Engineering Portfolio

Joshua Wu jo4467x@gmail.com

Background

This portfolio summarizes the more personalized long-term projects that I have worked on in my time in college. This portfolio does not include additional projects I have been a part of where I have not had a major role including my work in the 2 research labs I have been a part of as well as various other projects in classes. The projects listed below have all taken significant time to work on, usually over the course of several months.

I hope sees this will have an enjoyable time learning of the work that I have done for each of these projects as well as their final outcomes. Please feel free to contact me for more information on any of the projects listed below, or any additional questions regarding myself. My email address is included on the cover page.

Table of Contents:

ME2110 Robotics Competition

This was a class project in which we had to make autonomous robots to complete in a robotics competition.

Below is an overview of the game. There were a total of 5 tasks which once each was completed, would score you points depending on how well the task was achieved.

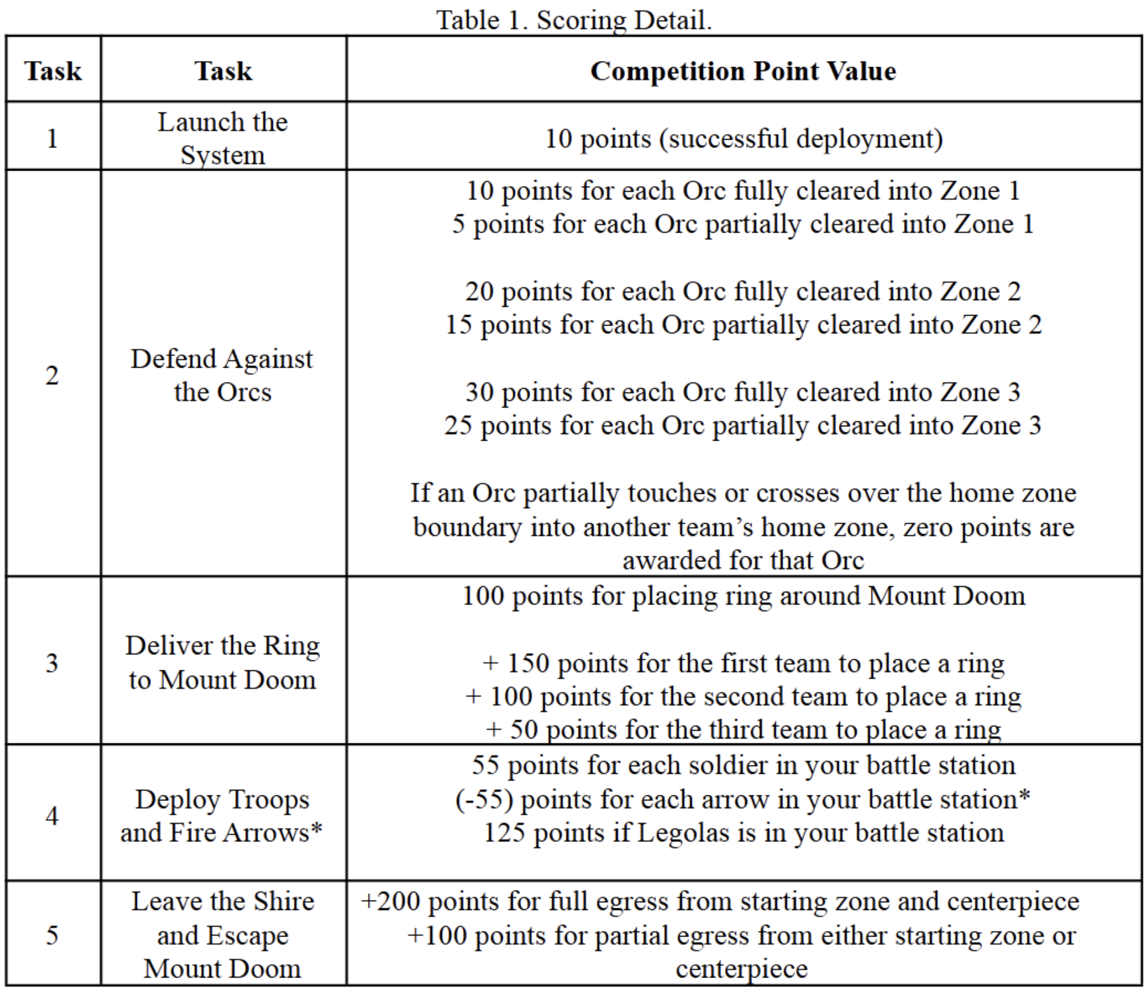


Table 1. Scoring Detail.

Task	Task	Competition Point Value
1	Launch the System	10 points (successful deployment)
2	Defend Against the Orcs	10 points for each Orc fully cleared into Zone 1 5 points for each Orc partially cleared into Zone 1 20 points for each Orc fully cleared into Zone 2 15 points for each Orc partially cleared into Zone 2 30 points for each Orc fully cleared into Zone 3 25 points for each Orc partially cleared into Zone 3 If an Orc partially touches or crosses over the home zone boundary into another team's home zone, zero points are awarded for that Orc
3	Deliver the Ring to Mount Doom	100 points for placing ring around Mount Doom + 150 points for the first team to place a ring + 100 points for the second team to place a ring + 50 points for the third team to place a ring
4	Deploy Troops and Fire Arrows*	55 points for each soldier in your battle station (-55) points for each arrow in your battle station* 125 points if Legolas is in your battle station
5	Leave the Shire and Escape Mount Doom	+200 points for full egress from starting zone and centerpiece +100 points for partial egress from either starting zone or centerpiece

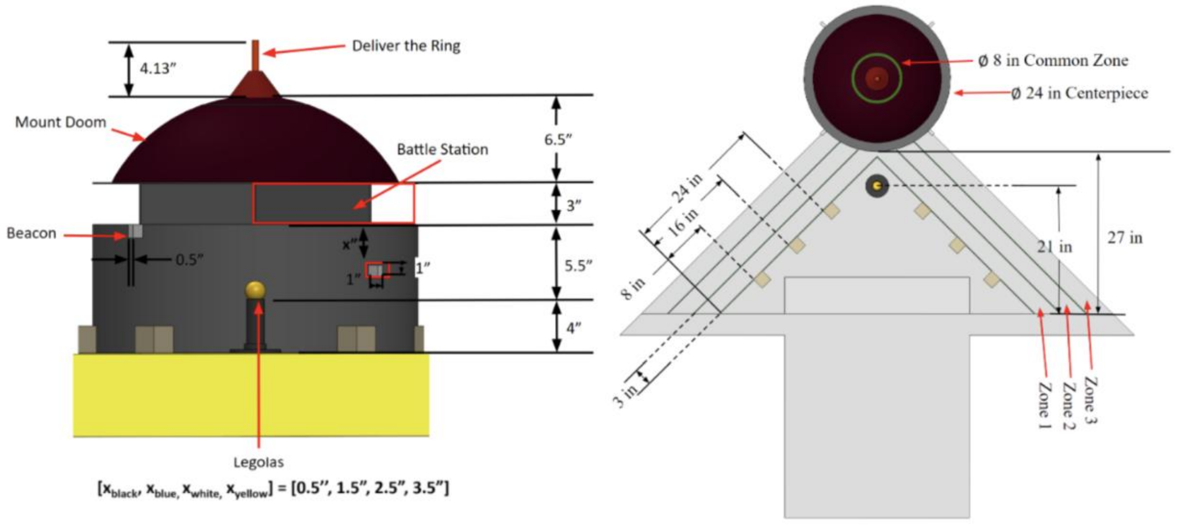


Figure 1: Overview of the Competition Field

Additionally, the competition limited the budget for the robot to be under $100, within size of 12x24x18 inches, as well as use only 2 DC motors, a pneumatic actuator, and 2 solenoids for electrically powered actuators.

Within the team, I was in charge of designing the soldier queue and the robot base. I also lead the fabrication portion of the robot. The development of the robot followed the standard engineering method. For each task, we identified the problem that needed to be solved. Then, we determined the requirements for the goal and the objectives that we wanted to reach for each problem in order for us to consider that we succeeded. For example, the ring delivery was timed so that the faster you were to put it on, the more points you would score, we decided that our target for that task would be 10 seconds. We then placed these requirements into a **house of quality** to determine the importance of each of them and developed a **specification sheet**. Adhering to the specification sheet, we made a **function tree** and came up with several designs to solve each function. Combining the different parts of the **morphological chart** together allowed us to come up with several final design ideas. We weighed each of our final designs against the requirements found in the house of quality and found our final design. Finally, prototypes were made, and through **iterative testing** we ended with a finished product.

As a group we decided to limit the use of motors as much as possible since we only had 2, so being in charge with the motor base I had to come up with ideas using as few motors as possible. Initially, I tried to make the base using a set of mousetraps which would activate and spring the robot out of the starting zone towards the center. However, this idea was eventually axed due to the unpredictability of the force within the mousetraps. In the end I decided to use one motor which would drive the back wheels of the robot for a 1-

dimensional movement from start to the center. Although this cost us a motor, we determined that it was worth it as robot positioning was the most important of the whole competition.

When it was my turn to design the soldier queue, both motors had already been used so my only options were to use a pneumatic actuator and/or a solenoid. I ended up using both the pneumatic actuator and a solenoid for the solution. The idea was the pneumatic actuator would control a flap that would start closed to prevent balls from rolling out of a pipe holding them. When the time was right, the pneumatics would activate releasing the flap and letting the balls roll into the correct zone. In addition to this however, we incorporated a solenoid as well which would start sitting slightly out of its socket blocking some additional balls which can be placed into enemy zones to subtract from their points. Once the mountain in the middle spun to an enemy zone, we would release the solenoid allowing these additional balls to roll in.

Our final design for the project was capable of scoring the max amount of point for a total of 960. However, in the final performance in the quarter finals round, our robot hit the center pole due to a misalignment issue in setup and as a result did not score any points. It was very unfortunate and had never happened during our practices.

Going forward, if I were to continue with this robot, I would have made a tool similar to a set square which could be used to help align the robot in setup to ensure the same starting position every single time. Additionally, our gears should have been 3d printed for higher precision and quality.

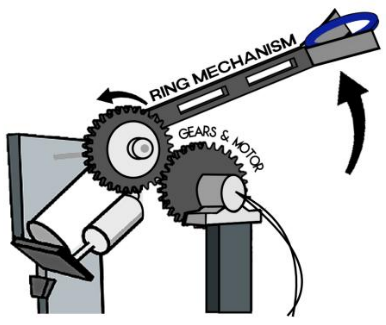


Figure 8 - Ring Delivery

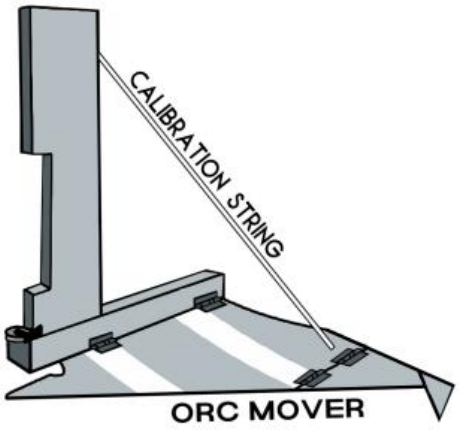


Figure 5 - Wing-like Attachments

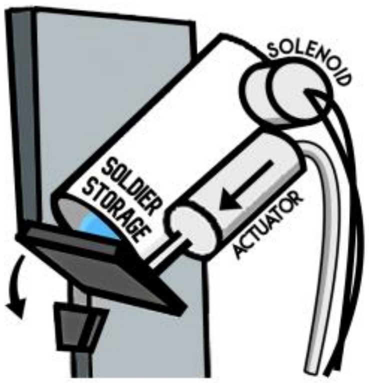


Figure 6 - Soldier Queue

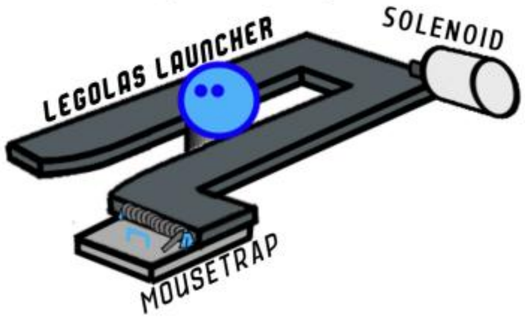


Figure 7 - Legolas Launcher

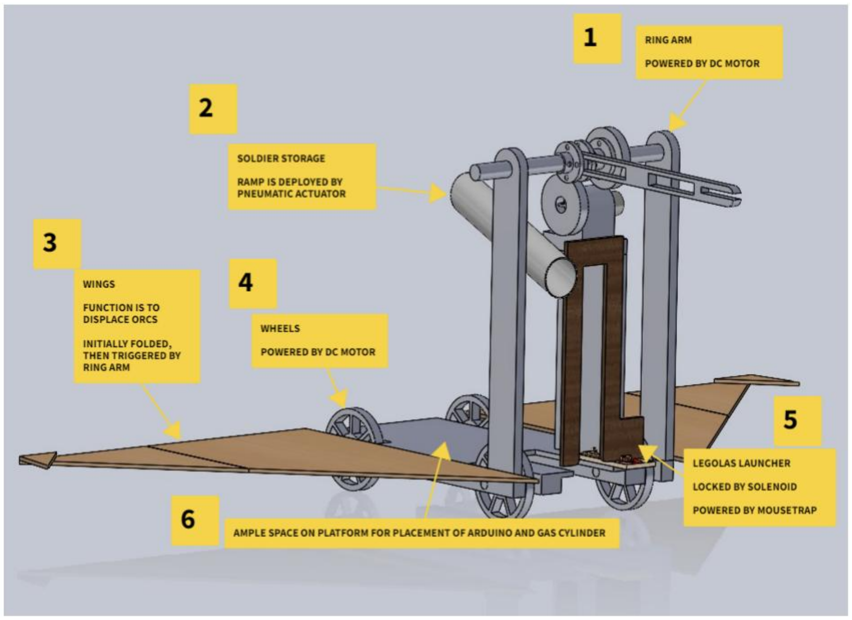

1
RING ARM
POWERED BY DC MOTOR
2
SOLDIER STORAGE
RAMP IS DEPLOYED BY PNEUMATIC ACTUATOR
3
WINGS
FUNCTION IS TO DISPLACE ORCS
INITIALLY FOLDED, THEN TRIGGERED BY RING ARM
4
WHEELS
POWERED BY DC MOTOR
5
LEGOLAS LAUNCHER
LOCKED BY SOLENOID
POWERED BY MOUSETRAP
6
AMPLE SPACE ON PLATFORM FOR PLACEMENT OF ARDUINO AND GAS CYLINDER

Figure 2: Final CAD of robot

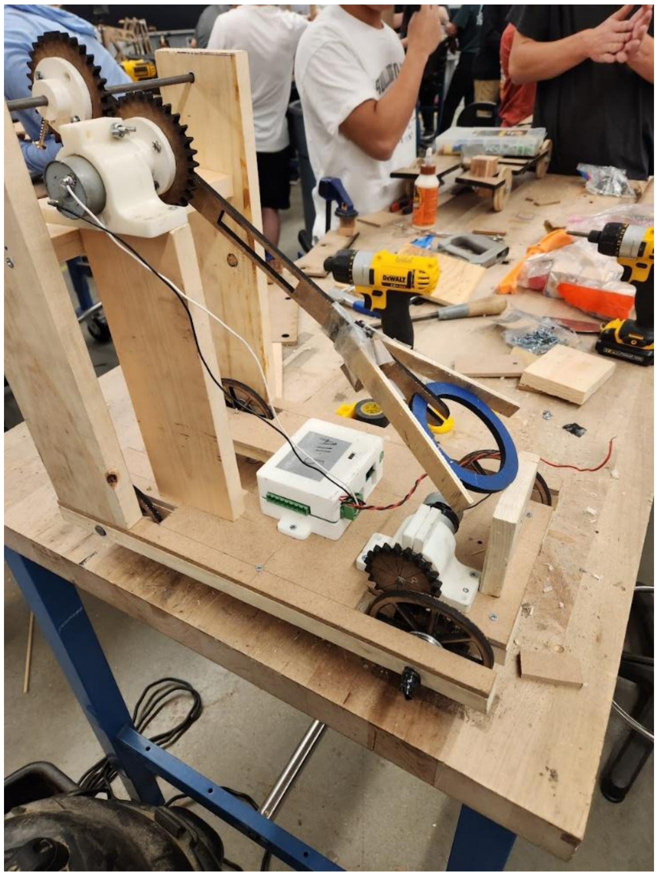


Figure 3: Picture of the Fabrication Process

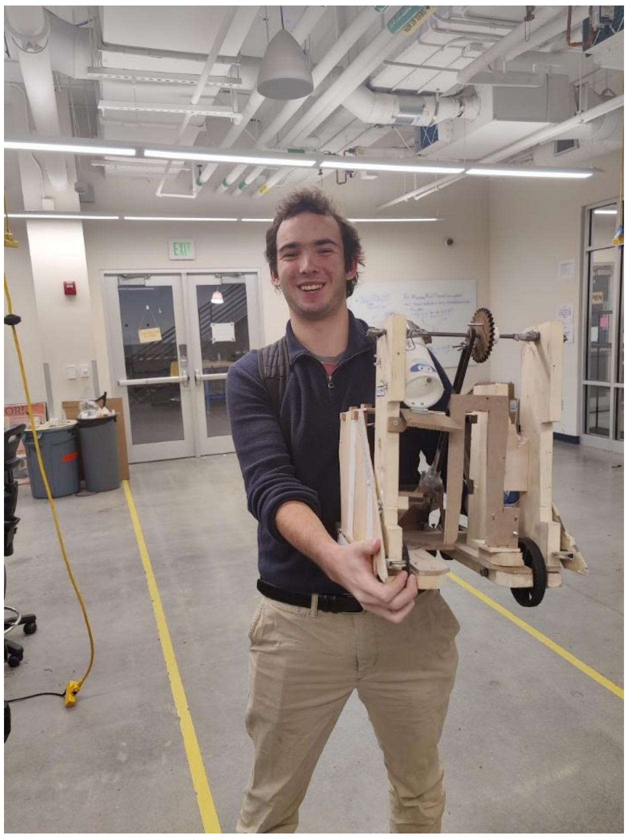


Figure 4: Finished Robot (My teammate is holding it here)

Table 2.1 - Morphological Chart (Launch System, and Legolas)

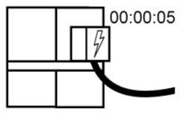
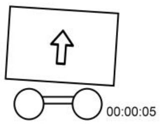
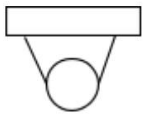
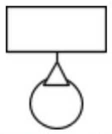
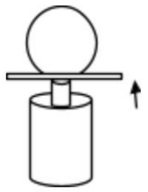
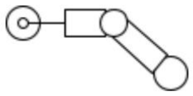
Function	Solution 1:	Solution 2:	Solution 3:
Launch System	 Timer countdown from when plugged in	Button press	 Sensor and timer when to start when box is lifted
Grab Legolas	 Grab From Sides	Lift with Fork	 Pick up from above
Lift Legolas	 Actuator	 Arm and System movement	Lift with motor and spool

Table 2.2 - Morphological Chart (Orcs, Soldiers, and Ring)

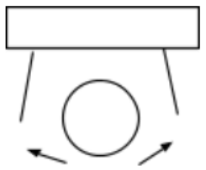
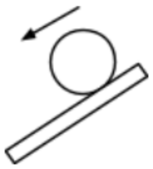
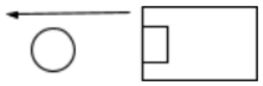
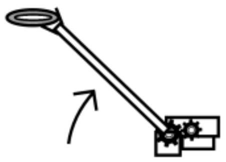
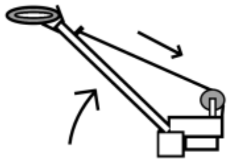
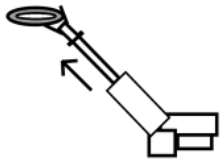
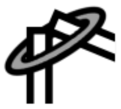
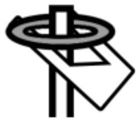
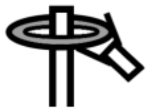
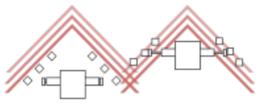
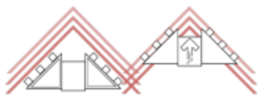
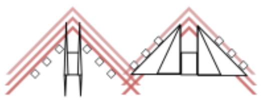
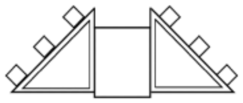
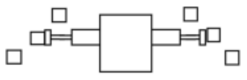
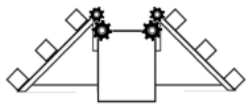
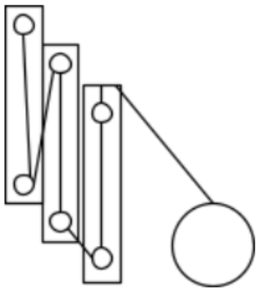
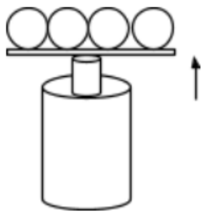
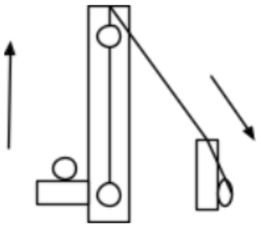
Function	Solution 1:	Solution 2:	Solution 3:
Place Legolas	 Release	 Roll off	 Launch horizontally
Raise Ring	 Motor attached to arm	 Spool of string	 Actuator
Place Ring	 Bar	 Fork	 Claw release
Push Orcs	 Actuator pumps at each orc location	 Sides of robot push orcs	 Sides of robot push orcs
Correct Zone	 Sides of robot shaped like wings	 Actuator pump distance limited	 Motor movement restricted
Raise Soldiers	 Vertical slide system	 Actuator	 Mousetrap

Table 2.3 - Morphological Chart (Drive System, and Soldiers)

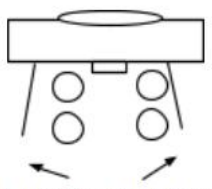
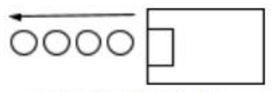
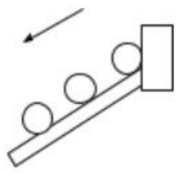
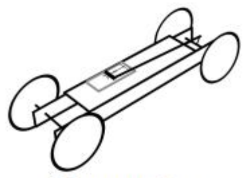
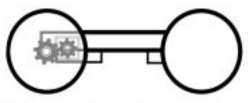
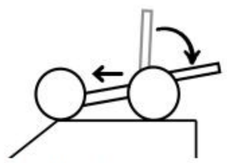
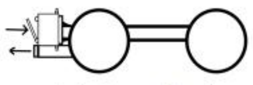
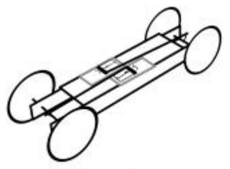
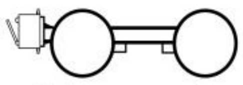
Function	Solution 1:	Solution 2:	Solution 3:
Identify Zone	Color sensor	Timer	Motion sensor
Place Soldiers Place Arrows	 Release from platform	 Shoot horizontally	 Slide down
Exit Shire	 Mousetrap Car	 Drive system from motors	 Shifting momentum
Retreat from Mt. Doom	 Actuator push w/ button/lever	 Mousetrap	 Drive system w/ button/lever

Table 3: Overview of our final costs

	Part Description	Material	Quantity	Units	Unit Cost	Cost
1	Body	MDF	216	Square Inch	$0.01	$2.16
2	Wings	MDF	112	Square Inch	$0.01	$1.12
3	Arm	MDF	20	Square Inch	$0.01	$0.20
4	Arm Mounts	Plywood	60	Square Inch	$0.05	$0.80
5	Motor Mounts	PLA	5	Cubic Inch	$0.82	$4.10
6	Hinges	Steel	8		$0.50	$4.00
7	Screws	Steel	40		$0.10	$4.00
8	Axles	Steel	30	Inch	$0.33	$5.00
9	Axle Mounts	PLA	3	Cubic Inch	$0.82	$2.46
10	Legolas Launcher	MDF	25	Square Inch	$0.01	$0.25
11	Wheels	MDF	80	Square Inch	$0.01	$0.80
12	Gears	MDF	20	Square Inch	$0.01	$0.20
13	Soldier Tube	PVC	8	Inch	$0.14	$1.13
14	Body	Plywood	150	Square Inch	$0.05	$7.50
Total Cost						$33.72

League of Legends Machine Learning Project

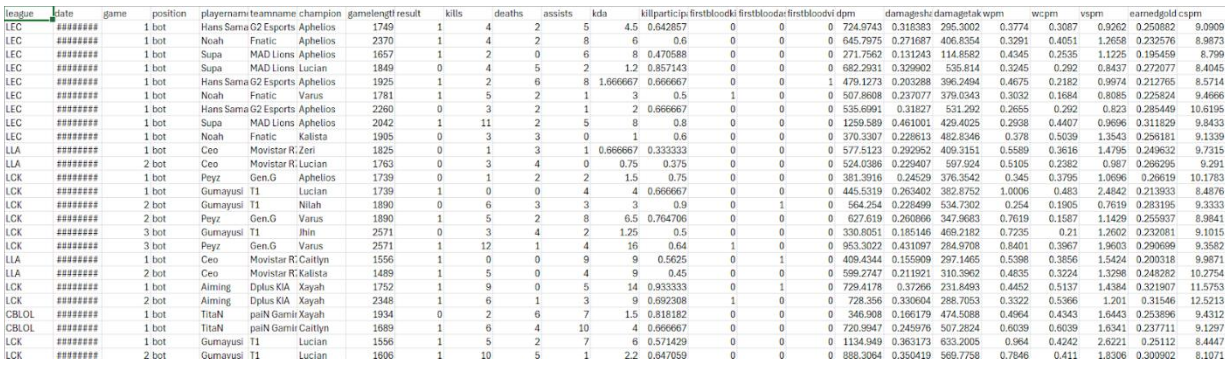
This was an in-class project where we focused on the video game League of Legends and the E-sports scene associated with it. Specifically, our goal was to predict which team would win the League of Legends 2024 World Championships with Machine Learning Methods using data taken from teams' previous match history.

Here is a short overview of the game:

The game revolves around two teams, each consisting of 5 players, who must siege towers and destroy the enemy's base ("nexus"). Each team is composed of 5 different roles, mostly corresponding to the lane they primarily play in at the beginning of the game: the toplaner, midlaner, botlaner. The players in the lanes are also commonly referred to as simply their lane's position, such as "top" or "bot."

Our approach to the project was first to determine which statistics for each role on the team were the most important in ensuring team victory. With this knowledge, we could then track each player on a team's overall performance over the season based on the metrics previously identified. And then using that information, we could rank teams based on how strong their players were for the season.

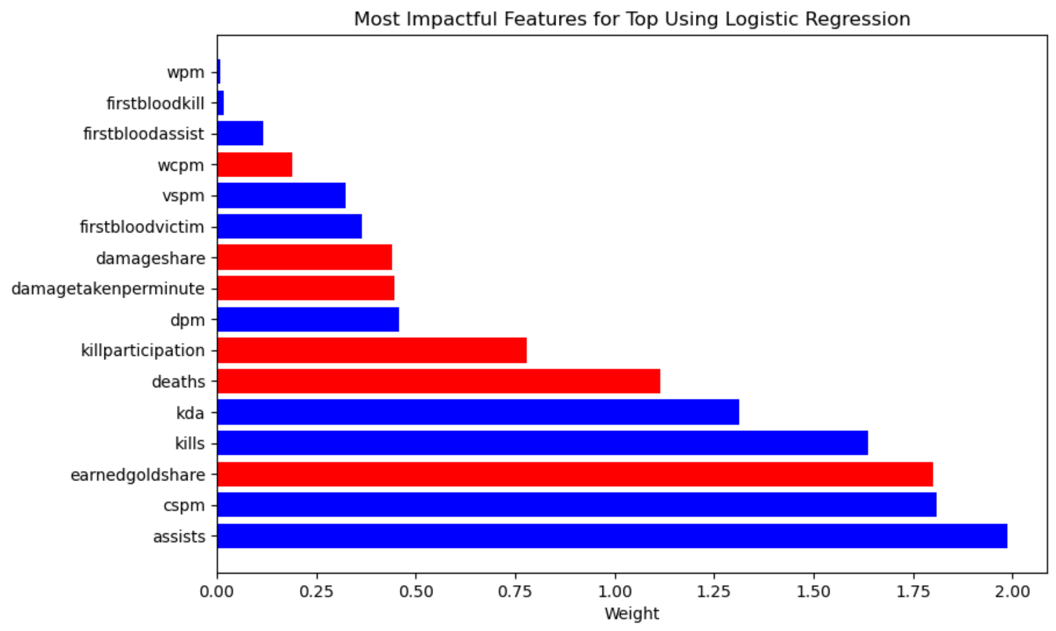
To start off, we first performed many data preprocessing methods to clean out our data. This involved **Data Cleaning**, in which we made python scripts to filter through all the excel data and remove any empty columns or rows. Next, we performed **feature engineering and selection**, in which we created new features from our data that would encompass and represent multiple other statistics and also removed features determined to be unimportant in our research. This cut our spreadsheet size down and simplified it to make it easier to analyze. After this, we split the data into 5 separate datasets, one for each role in the game. This enabled us to run our model on each role individually so we would be able to draw different conclusions for each role.

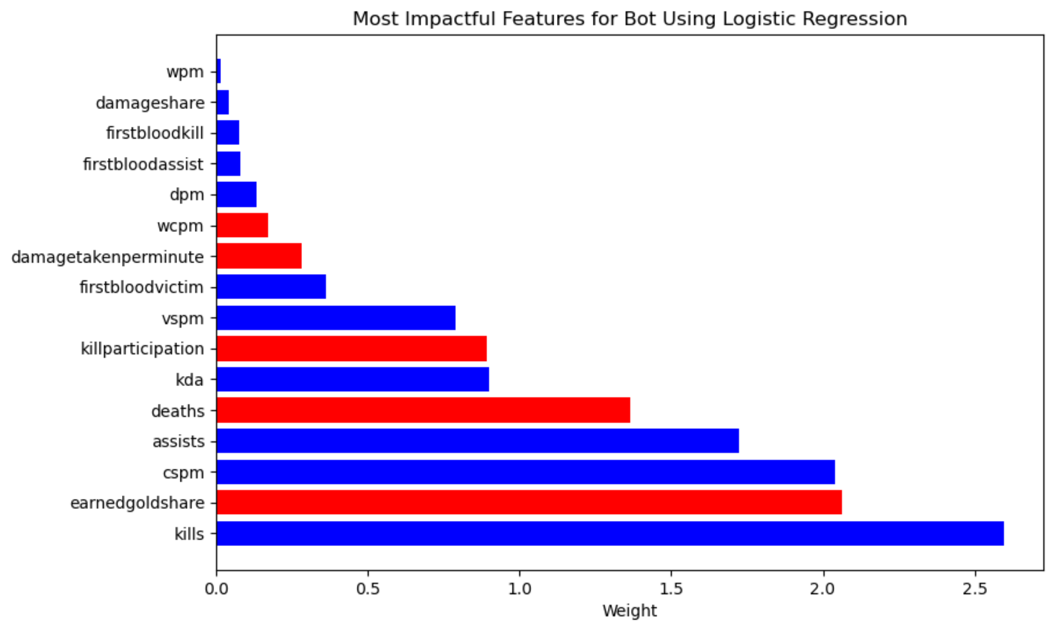


league	date	game	position	playernam	teamname	champion	gamelengt	result	kills	deaths	assists	kda	killparticip	firstbloodki	firstblooda	firstbloodvi	dpm	damagesha	damagetak	wpm	wcpm	vspm	earnedgold	cspm
LEC	########	1	bot	Hans Sama	G2 Esports	Aphelios	1749	1	4	2	5	4.5	0.642857	0	0	0	724.9743	0.318383	295.3002	0.3774	0.3087	0.9262	0.250882	9.0909
LEC	########	1	bot	Noah	Fnatic	Aphelios	2370	1	4	2	8	6	0.6	0	0	0	645.7975	0.271687	406.8354	0.3291	0.4051	1.2658	0.232576	8.9873
LEC	########	1	bot	Supa	MAD Lions	Aphelios	1657	1	2	0	6	8	0.470588	0	0	0	271.7562	0.131243	114.8582	0.4345	0.2535	1.1225	0.195459	8.799
LEC	########	1	bot	Supa	MAD Lions	Lucian	1849	0	4	5	2	1.2	0.857143	0	0	0	682.2931	0.329902	535.814	0.3245	0.292	0.8437	0.272077	8.4045
LEC	########	1	bot	Hans Sama	G2 Esports	Aphelios	1925	1	2	6	8	1.666667	0.666667	0	0	1	479.1273	0.203288	396.2494	0.4675	0.2182	0.9974	0.212765	8.5714
LEC	########	1	bot	Noah	Fnatic	Varus	1781	1	5	2	1	3	0.5	1	0	0	507.8608	0.237077	379.0343	0.3032	0.1684	0.8085	0.225824	9.4666
LEC	########	1	bot	Hans Sama	G2 Esports	Aphelios	2260	0	3	2	1	2	0.666667	0	0	0	535.6991	0.31827	531.292	0.2655	0.292	0.823	0.285449	10.6195
LEC	########	1	bot	Supa	MAD Lions	Aphelios	2042	1	11	2	5	8	0.8	0	0	0	1259.589	0.461001	429.4025	0.2938	0.4407	0.9696	0.311829	9.8433
LEC	########	1	bot	Noah	Fnatic	Kalista	1905	0	3	3	0	1	0.6	0	0	0	370.3307	0.228613	482.8346	0.378	0.5039	1.3543	0.256181	9.1339
LLA	########	1	bot	Ceo	Movistar R	Zeri	1825	0	1	3	1	0.666667	0.333333	0	0	0	577.5123	0.292952	409.3151	0.5589	0.3616	1.4795	0.249632	9.7315
LLA	########	2	bot	Ceo	Movistar R	Lucian	1763	0	3	4	0	0.75	0.375	0	0	0	524.0386	0.229407	597.924	0.5105	0.2382	0.987	0.266295	9.291
LCK	########	1	bot	Peyz	Gen.G	Aphelios	1739	0	1	2	2	1.5	0.75	0	0	0	381.3916	0.24529	376.3542	0.345	0.3795	1.0696	0.26619	10.1783
LCK	########	1	bot	Gumayusi	T1	Lucian	1739	1	0	0	4	4	0.666667	0	0	0	445.5319	0.263402	382.8752	1.0006	0.483	2.4842	0.213933	8.4876
LCK	########	2	bot	Gumayusi	T1	Nilah	1890	0	6	3	3	3	0.9	0	1	0	564.254	0.228499	534.7302	0.254	0.1905	0.7619	0.283195	9.3333
LCK	########	2	bot	Peyz	Gen.G	Varus	1890	1	5	2	8	6.5	0.764706	0	0	0	627.619	0.260866	347.9683	0.7619	0.1587	1.1429	0.255937	8.9841
LCK	########	3	bot	Gumayusi	T1	Jhin	2571	0	3	4	2	1.25	0.5	0	0	0	330.8051	0.185146	469.2182	0.7235	0.21	1.2602	0.232081	9.1015
LCK	########	3	bot	Peyz	Gen.G	Varus	2571	1	12	1	4	16	0.64	1	0	0	953.3022	0.431097	284.9708	0.8401	0.3967	1.9603	0.290699	9.3582
LLA	########	1	bot	Ceo	Movistar R	Caitlyn	1556	1	0	0	9	9	0.5625	0	1	0	409.4344	0.155909	297.1465	0.5398	0.3856	1.5424	0.200318	9.9871
LLA	########	2	bot	Ceo	Movistar R	Kalista	1489	1	5	0	4	9	0.45	0	0	0	599.2747	0.211921	310.3962	0.4835	0.3224	1.3298	0.248282	10.2754
LCK	########	1	bot	Aiming	Dplus KIA	Xayah	1752	1	9	0	5	14	0.933333	0	1	0	729.4178	0.37266	231.8493	0.4452	0.5137	1.4384	0.321907	11.5753
LCK	########	2	bot	Aiming	Dplus KIA	Xayah	2348	1	6	1	3	9	0.692308	1	0	0	728.356	0.330604	288.7053	0.3322	0.5366	1.201	0.31546	12.5213
CBLOL	########	1	bot	TitaN	paiN Gamir	Xayah	1934	0	2	6	7	1.5	0.818182	0	0	0	346.908	0.166179	474.5088	0.4964	0.4343	1.6443	0.253896	9.4312
CBLOL	########	1	bot	TitaN	paiN Gamir	Caitlyn	1689	1	6	4	10	4	0.666667	0	0	0	720.9947	0.245976	507.2824	0.6039	0.6039	1.6341	0.237711	9.1297
LCK	########	1	bot	Gumayusi	T1	Lucian	1556	1	5	2	7	6	0.571429	0	0	0	1134.949	0.363173	633.2005	0.964	0.4242	2.6221	0.25112	8.4447
LCK	########	2	bot	Gumayusi	T1	Lucian	1606	1	10	5	1	2.2	0.647059	0	0	0	888.3064	0.350419	569.7758	0.7846	0.411	1.8306	0.300902	8.1071

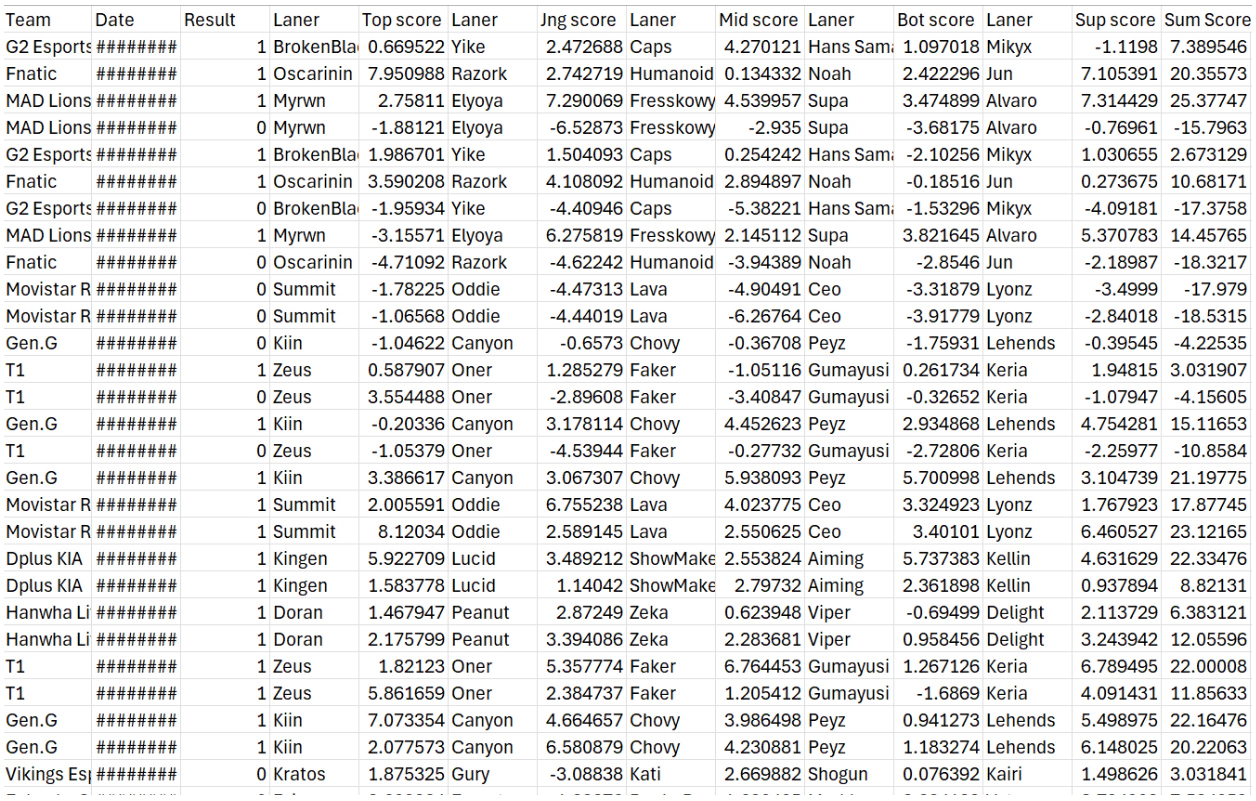
Figure: Example Dataset

Our first model we implemented was **Logistic Regression**. We used this to gain coefficients to attach to each of our features to determine which ones were most important in helping a player in each role win (1) or lose (0) a game. Below are 2 examples of what came out of this model.





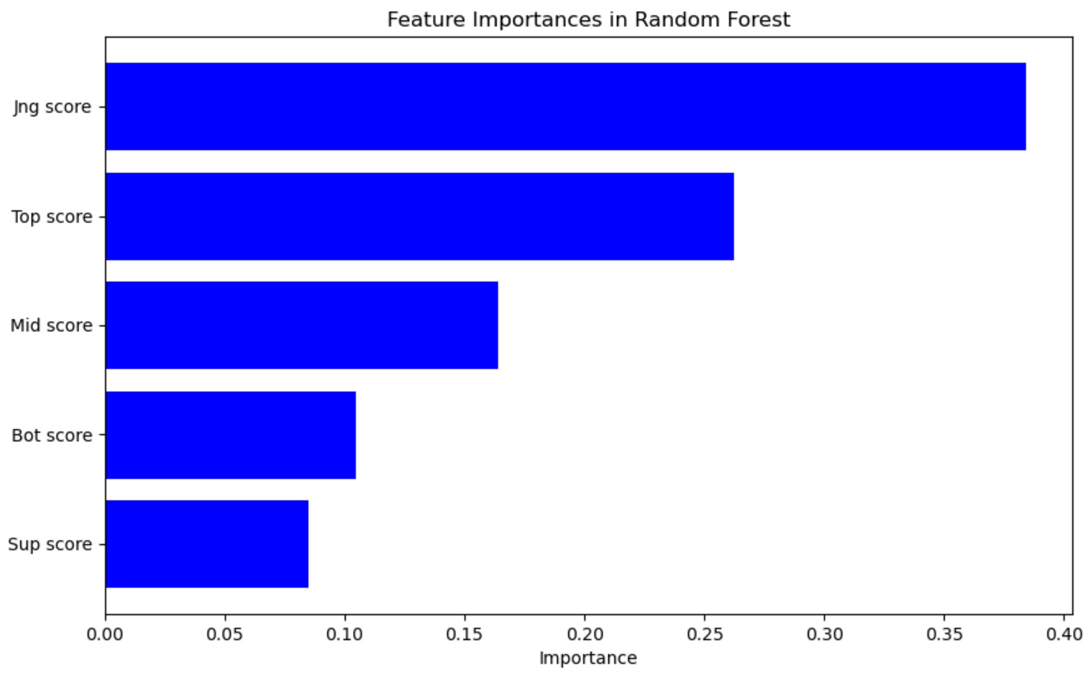
In the next step of the project, we took the weights for each feature and multiplied them by the values of the feature to assign each feature a value. By then summing up each feature value for each player, we were able to get a score for a player in every game they played in year. We then placed each player and their score together with each other player on their team for every game played in the year. This new dataset would allow us to track team performance in games.



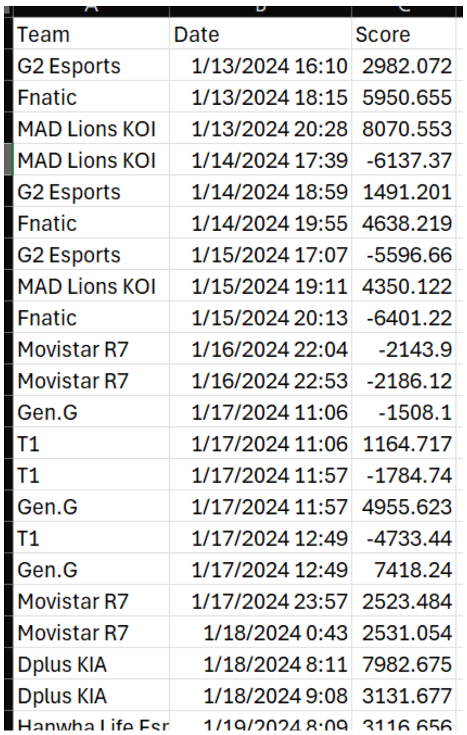
Team	Date	Result	Laner	Top score	Laner	Jng score	Laner	Mid score	Laner	Bot score	Laner	Sup score	Sum Score
G2 Esports	########	1	BrokenBla	0.669522	Yike	2.472688	Caps	4.270121	Hans Sama	1.097018	Mikyx	-1.1198	7.389546
Fnatic	########	1	Oscarinin	7.950988	Razork	2.742719	Humanoid	0.134332	Noah	2.422296	Jun	7.105391	20.35573
MAD Lions	########	1	Myrwn	2.75811	Elyoya	7.290069	Fresskowy	4.539957	Supa	3.474899	Alvaro	7.314429	25.37747
MAD Lions	########	0	Myrwn	-1.88121	Elyoya	-6.52873	Fresskowy	-2.935	Supa	-3.68175	Alvaro	-0.76961	-15.7963
G2 Esports	########	1	BrokenBla	1.986701	Yike	1.504093	Caps	0.254242	Hans Sama	-2.10256	Mikyx	1.030655	2.673129
Fnatic	########	1	Oscarinin	3.590208	Razork	4.108092	Humanoid	2.894897	Noah	-0.18516	Jun	0.273675	10.68171
G2 Esports	########	0	BrokenBla	-1.95934	Yike	-4.40946	Caps	-5.38221	Hans Sama	-1.53296	Mikyx	-4.09181	-17.3758
MAD Lions	########	1	Myrwn	-3.15571	Elyoya	6.275819	Fresskowy	2.145112	Supa	3.821645	Alvaro	5.370783	14.45765
Fnatic	########	0	Oscarinin	-4.71092	Razork	-4.62242	Humanoid	-3.94389	Noah	-2.8546	Jun	-2.18987	-18.3217
Movistar R	########	0	Summit	-1.78225	Oddie	-4.47313	Lava	-4.90491	Ceo	-3.31879	Lyonz	-3.4999	-17.979
Movistar R	########	0	Summit	-1.06568	Oddie	-4.44019	Lava	-6.26764	Ceo	-3.91779	Lyonz	-2.84018	-18.5315
Gen.G	########	0	Kiin	-1.04622	Canyon	-0.6573	Chovy	-0.36708	Peyz	-1.75931	Lehends	-0.39545	-4.22535
T1	########	1	Zeus	0.587907	Oner	1.285279	Faker	-1.05116	Gumayusi	0.261734	Keria	1.94815	3.031907
T1	########	0	Zeus	3.554488	Oner	-2.89608	Faker	-3.40847	Gumayusi	-0.32652	Keria	-1.07947	-4.15605
Gen.G	########	1	Kiin	-0.20336	Canyon	3.178114	Chovy	4.452623	Peyz	2.934868	Lehends	4.754281	15.11653
T1	########	0	Zeus	-1.05379	Oner	-4.53944	Faker	-0.27732	Gumayusi	-2.72806	Keria	-2.25977	-10.8584
Gen.G	########	1	Kiin	3.386617	Canyon	3.067307	Chovy	5.938093	Peyz	5.700998	Lehends	3.104739	21.19775
Movistar R	########	1	Summit	2.005591	Oddie	6.755238	Lava	4.023775	Ceo	3.324923	Lyonz	1.767923	17.87745
Movistar R	########	1	Summit	8.12034	Oddie	2.589145	Lava	2.550625	Ceo	3.40101	Lyonz	6.460527	23.12165
Dplus KIA	########	1	Kingen	5.922709	Lucid	3.489212	ShowMake	2.553824	Aiming	5.737383	Kellin	4.631629	22.33476
Dplus KIA	########	1	Kingen	1.583778	Lucid	1.14042	ShowMake	2.79732	Aiming	2.361898	Kellin	0.937894	8.82131
Hanwha Li	########	1	Doran	1.467947	Peanut	2.87249	Zeka	0.623948	Viper	-0.69499	Delight	2.113729	6.383121
Hanwha Li	########	1	Doran	2.175799	Peanut	3.394086	Zeka	2.283681	Viper	0.958456	Delight	3.243942	12.05596
T1	########	1	Zeus	1.82123	Oner	5.357774	Faker	6.764453	Gumayusi	1.267126	Keria	6.789495	22.00008
T1	########	1	Zeus	5.861659	Oner	2.384737	Faker	1.205412	Gumayusi	-1.6869	Keria	4.091431	11.85633
Gen.G	########	1	Kiin	7.073354	Canyon	4.664657	Chovy	3.986498	Peyz	0.941273	Lehends	5.498975	22.16476
Gen.G	########	1	Kiin	2.077573	Canyon	6.580879	Chovy	4.230881	Peyz	1.183274	Lehends	6.148025	20.22063
Vikings Es	########	0	Kratos	1.875325	Gury	-3.08838	Kati	2.669882	Shogun	0.076392	Kairi	1.498626	3.031841

Figure: Example of New Dataset

With this data, we applied a **random forest** model to it with the goal of determining which role in a team has the greatest impact in determining victory. The random forest model was preferred in this case because the model is able to deal with complex, non-linear relationships which logistic and linear regression can not do. We hypothesized that relationships between player roles would be a lot more complex in determining victory compared to simple stats like gold advantage or number of kills. Our final results for the section are below:

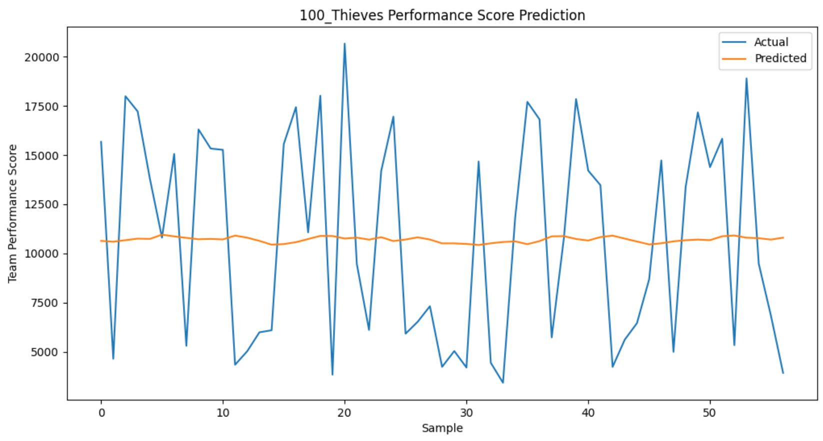


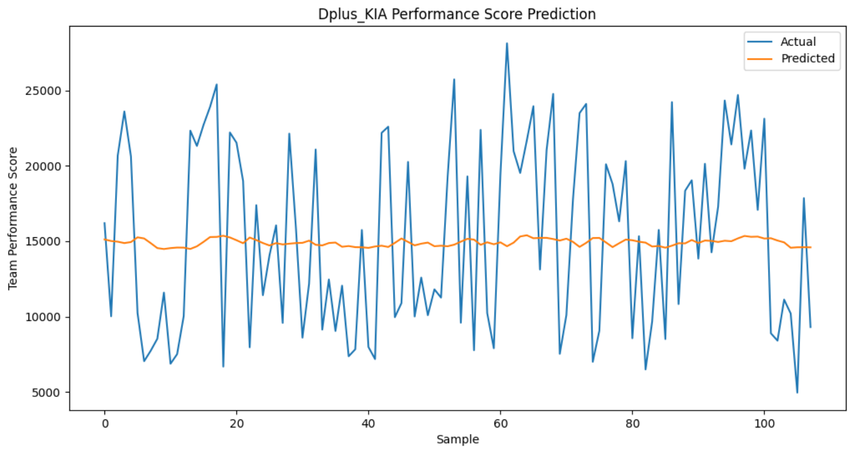
With a score of how well a player performs in a game, and a measure of how important a role is in determining the outcome of a game, the overall impact of a player on a team in a game could be determined. By then summing player impacts in a team, a team's overall strength could be determined per game.



Team	Date	Score
G2 Esports	1/13/2024 16:10	2982.072
Fnatic	1/13/2024 18:15	5950.655
MAD Lions KOI	1/13/2024 20:28	8070.553
MAD Lions KOI	1/14/2024 17:39	-6137.37
G2 Esports	1/14/2024 18:59	1491.201
Fnatic	1/14/2024 19:55	4638.219
G2 Esports	1/15/2024 17:07	-5596.66
MAD Lions KOI	1/15/2024 19:11	4350.122
Fnatic	1/15/2024 20:13	-6401.22
Movistar R7	1/16/2024 22:04	-2143.9
Movistar R7	1/16/2024 22:53	-2186.12
Gen.G	1/17/2024 11:06	-1508.1
T1	1/17/2024 11:06	1164.717
T1	1/17/2024 11:57	-1784.74
Gen.G	1/17/2024 11:57	4955.623
T1	1/17/2024 12:49	-4733.44
Gen.G	1/17/2024 12:49	7418.24
Movistar R7	1/17/2024 23:57	2523.484
Movistar R7	1/18/2024 0:43	2531.054
Dplus KIA	1/18/2024 8:11	7982.675
Dplus KIA	1/18/2024 9:08	3131.677
Hanwha Life Esr	1/19/2024 8:09	3116.656

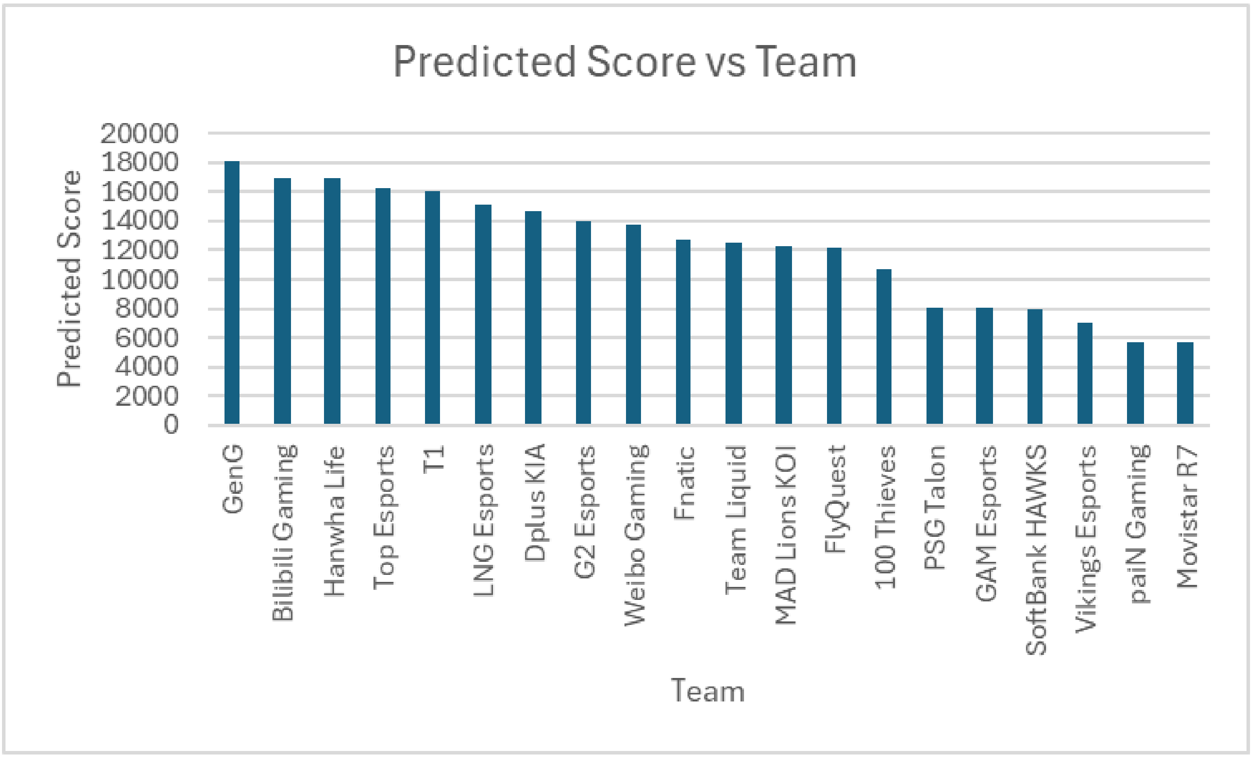
With this new data, we performed one final ML model which was an **LSTM**. We planned to track each team's performance over the time of one year to determine trends such as improving or declining performance leading up to international competition. Below are some of our final results:



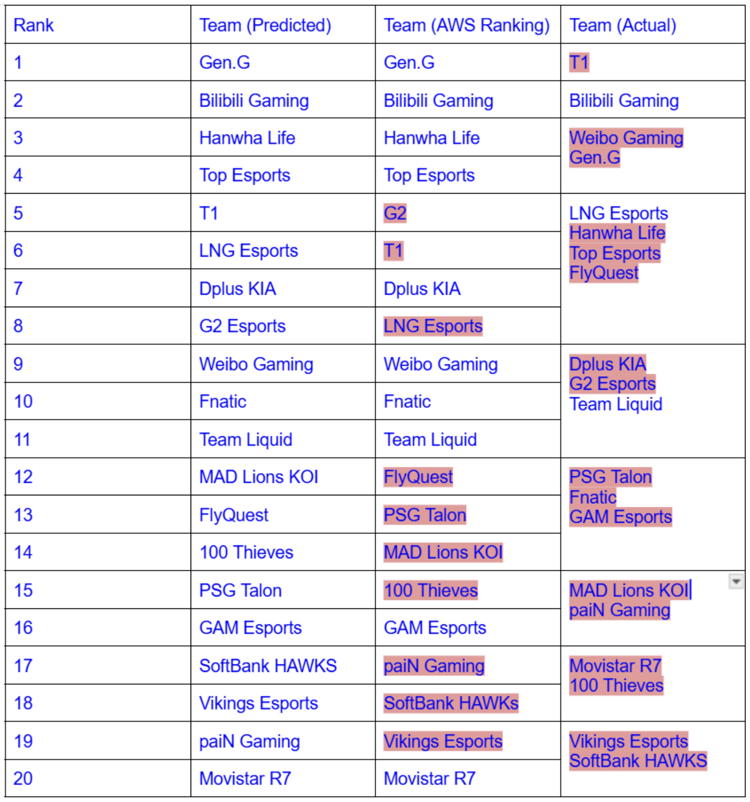


The results from this last model we implemented were underwhelming as no team had any recognizable trend in their game performances leading up to worlds. Many teams were shown to be in slumps or booms during periods within the year, but had no noticeable average improvements or declines over the year. As a result of the consistently oscillating data for each team, the LSTM was unable to find any patterns and therefore could not make any substantial predictions. The final predicted team performance scores differed little from the team performance score averaged over the entire year.

Our final rankings for each team for their predicted performance as worlds were as follows:



Comparing this to the predictions made by official analysts and the true final results:



Rank	Team (Predicted)	Team (AWS Ranking)	Team (Actual)
1	Gen.G	Gen.G	T1
2	Bilibili Gaming	Bilibili Gaming	Bilibili Gaming
3	Hanwha Life	Hanwha Life	Weibo Gaming Gen.G
4	Top Esports	Top Esports	
5	T1	G2	LNG Esports Hanwha Life Top Esports FlyQuest
6	LNG Esports	T1	
7	Dplus KIA	Dplus KIA	
8	G2 Esports	LNG Esports	
9	Weibo Gaming	Weibo Gaming	Dplus KIA G2 Esports Team Liquid
10	Fnatic	Fnatic	
11	Team Liquid	Team Liquid	
12	MAD Lions KOI	FlyQuest	PSG Talon Fnatic GAM Esports
13	FlyQuest	PSG Talon	
14	100 Thieves	MAD Lions KOI	
15	PSG Talon	100 Thieves	MAD Lions KOI paiN Gaming
16	GAM Esports	GAM Esports	
17	SoftBank HAWKS	paiN Gaming	Movistar R7 100 Thieves
18	Vikings Esports	SoftBank HAWKs	
19	paiN Gaming	Vikings Esports	Vikings Esports SoftBank HAWKS
20	Movistar R7	Movistar R7	

Overall we were very impressed with our results, as we were able to obtain very similar predictions to the ones made by the analysts. Our final standard deviation between the official final results was 2.5 and the analyst standard deviation was 2.429, making us almost equal. Apart from the LSTM model, our other 2 models implemented proved to be extremely accurate. The logistic regression model was able to predict the correct outcome of a game 92.79% and the random forest model was able to predict the correct outcome 97% of the time.

For the next steps of this project, I would love to develop an additional model to track team performance with respect to the region they are from. One big failure in our model was that we did not account for the fact that different regions have teams of different skill levels. So if teams in a weaker region play against each other, their performance scores would be inflated. To account for this in our current model, we took from online a ranking of each region and their international performance. We multiplied this with our results to get our final rankings. Being able to have a model for this ourselves would greatly improve the quality of our results.

Automatic Card Shuffler (In Progress)

This started off as a personal project of mine which I eventually evolved to be my capstone project. Initially I started off as wanting to make a card shuffler that would be capable of shuffling my Magic: The Gathering (Trading Card Game) decks. Shuffling decks of 100 cards by hand when all the cards are sleeved proved to be a challenge, so I wanted a system that would be able to do it automatically.

Using parts that I had from a previous mechatronics class, I took some motors and wheels off a robot, took a stm32 Nucleo microcontroller and an ultrasonic sensor and made a first prototype. A link to the video can be found here: https://youtu.be/cs_fdS0fooY.

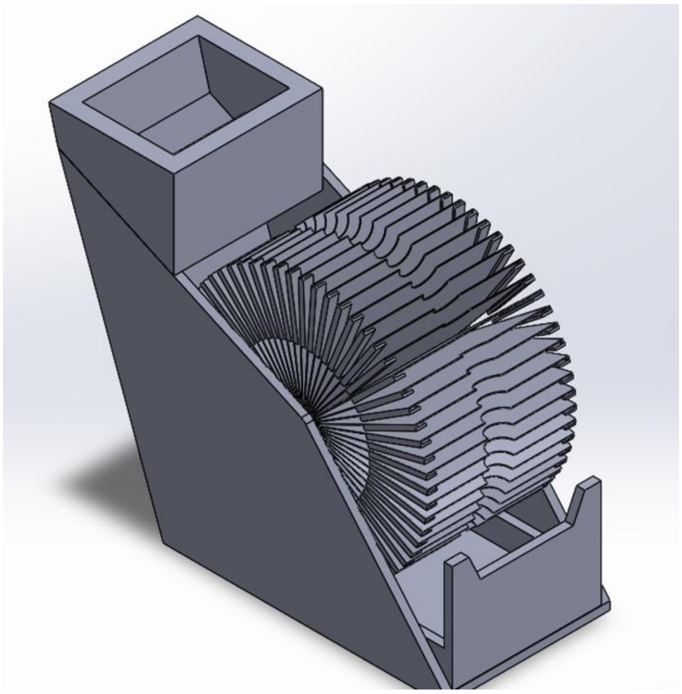
The system is based on a very simple machine already on the market which can be purchased from amazon for $20. What made my machine functionally different however, was its ability to shuffle cards of a much larger size, accounting for sleeved cards. There is nothing on the market that currently supports this. Additionally, for a more automated experience, the ultrasonic sensor in place detects if cards have been put into the machine and can also detect when all the cards have been fed out. In this situation, the machine automatically turns off with no need for human interaction.

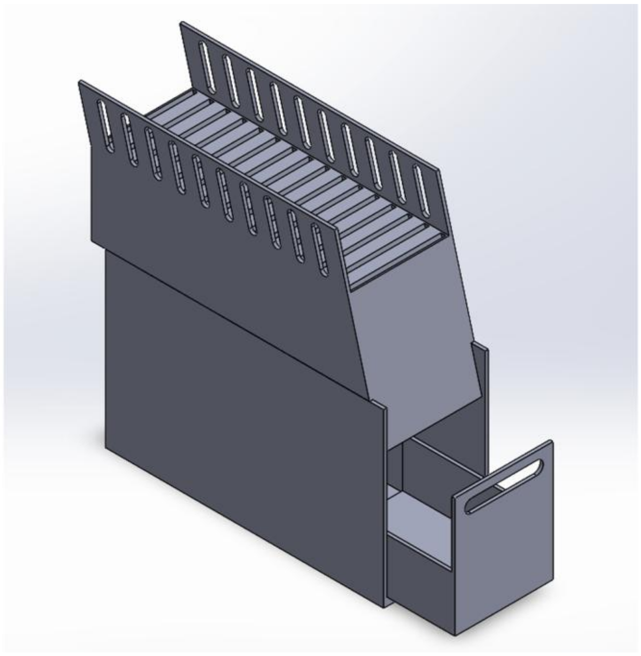
Additionally in this prototype system, I implemented an RGB color sensor which was able to detect the color of the back of all the cards being shuffled. I originally had this because I was afraid if someone put in cards that were unsleeved, the machine could damage it. The RGB sensor ran on i2c protocol and was able to detect if a card was sleeved or not. In the case where the RGB sensor detected a brown color (original color of the back) the motors would slow down and shuffle slower, making it safer for the cards.

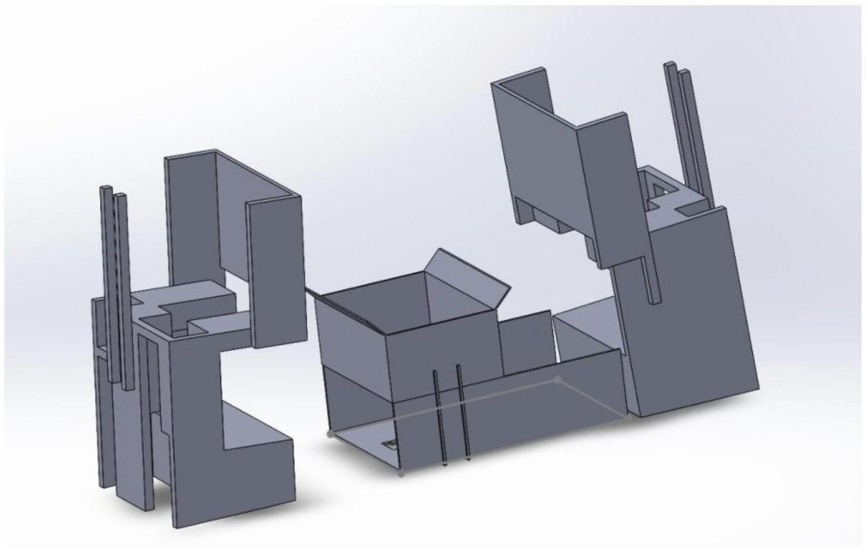
The entire system was controlled using an STM32 Microcontroller programmed on the STM32 IDE using C.

After having made the prototype, I had several ideas for improvements, so when the time for my capstone project came, I decided to work on the shuffler as my project. In this instance however, my new goal was to make a Universal, Automated Card Shuffler. One key aspect this project was to differ with my original project was that I didn’t just want the machine to automatically shuffle my cards, I wanted it to be capable of shuffling ANY card from ANY card game, sleeved or unsleeved. With the new project, my team and I did the due diligence of sending out surveys into local trading card game groups in hopes of finding more important customer requirements that we would need for a product. What came up most in this survey was the safety of the cards, the speed of shuffling, the randomness of the shuffle, and size. Therefore, we seeked to work on these aspects first and foremost in this second iteration of the card shuffler.

Following the engineering method, we evaluated each of the requirements on their importance and generated engineering specifications using tools like the House of Quality. Using function trees and morphological charts, we ended up with 3 final designs to choose from:



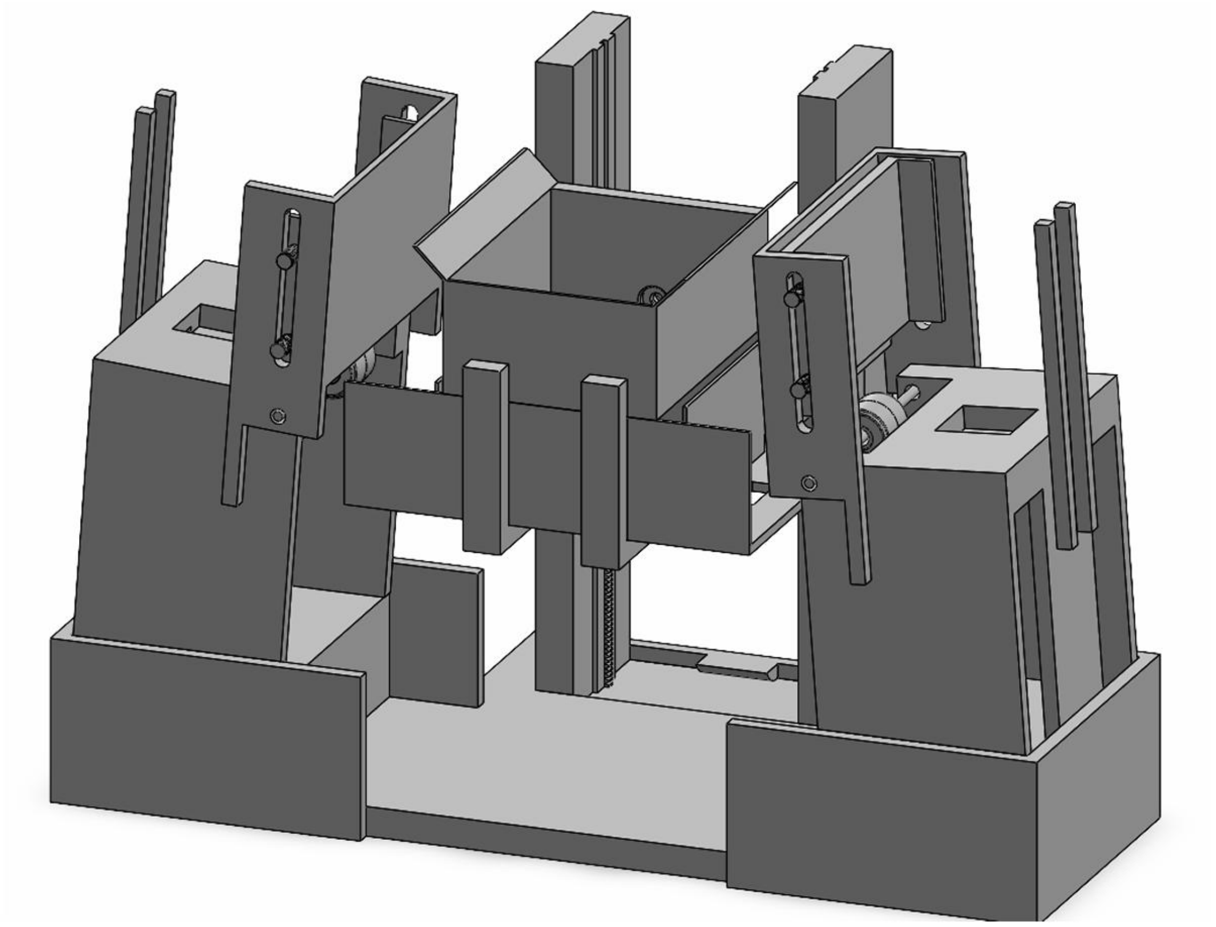




One resembled a system similar to how card shufflers at vegas work, where a random seed is inputted into a wheel which dispenses cards fed into it accordingly. Another was a completely new system which worked by splitting cards into several small piles and merging the piles together. The third system was similar to my original prototype. After evaluating the 3 designs, we ended up choosing the 3rd design as we also needed to consider price of manufacturing and this design was much cheaper to make compared to the wheel and also allowed for a much more random shuffle than the second design.

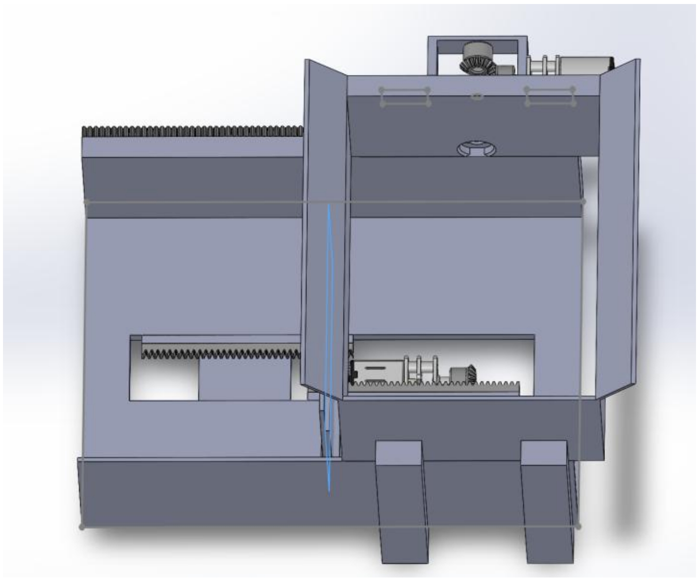
One key aspect that this new design improved on from my prototype was its randomness. In the original prototype the card shuffler can be approximated to model a singular riffle shuffle. A singular riffle shuffle represents a very poor randomness and casinos usually require at least seven consecutive riffle shuffles to assure perfect randomness. In the original prototype this would simply be achieved by taking the completed pile and reseparating it into 2 piles to be shuffled again. The new design featured a system that would allow for automatic recirculation of cards. Cards would pile into the middle and be automatically transported back to the top, split into 2 piles and reshuffled again.

Below is a CAD of our final design:

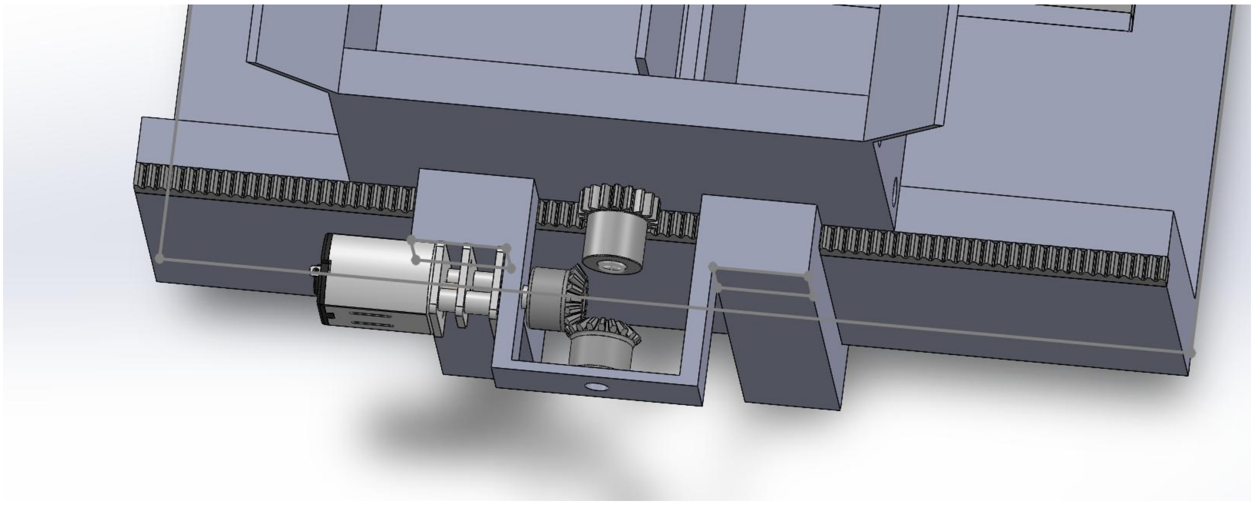


My contributions to this specific design include the tower system with the motors and the sensors, the central catcher which takes the cards in, the splitter which divides the cards into 2 piles, and the pusher which pushes the 2 piles back into their respective towers.

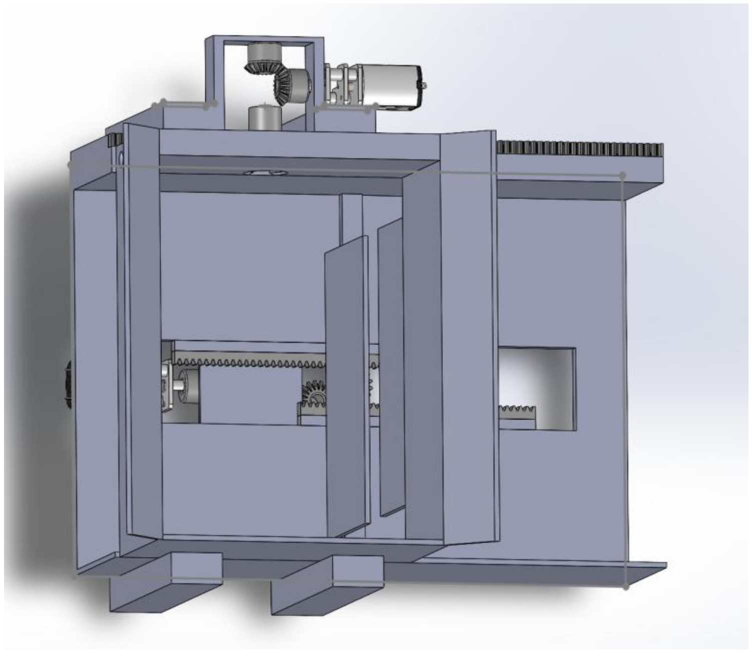
The new splitter system works by initially starting in this position:



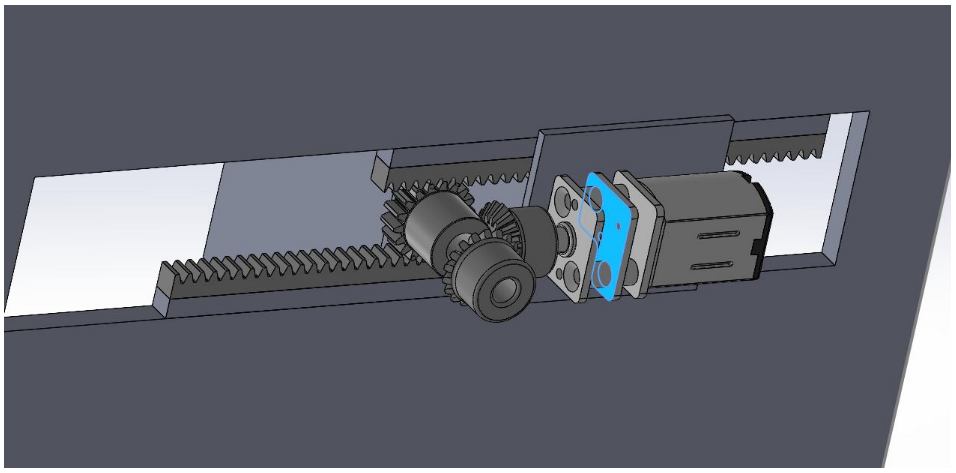
Cards fall through the top catcher into one side of the splitter, then once all the cards are piled in this section, half the cards will be sitting on the sliding mechanism and the other half will still be in the catcher. Then, using a gear rack system on the back,

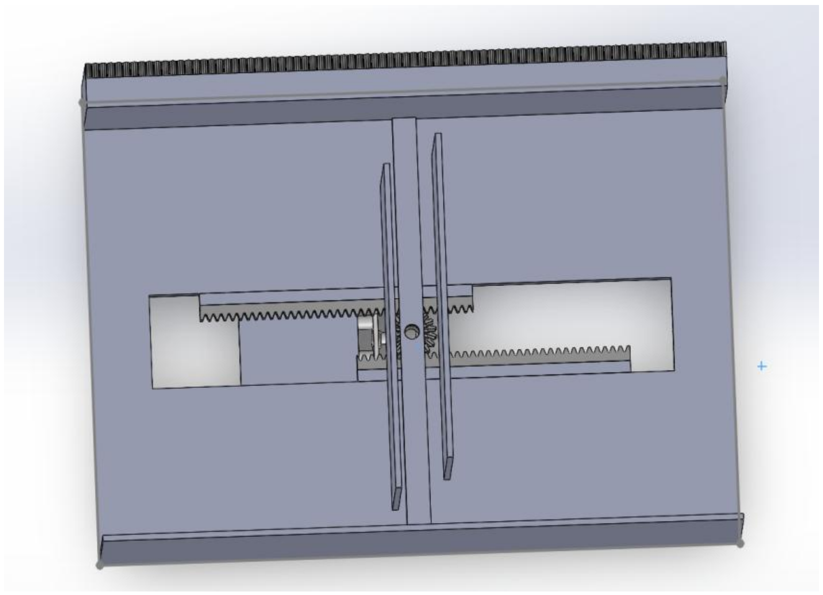


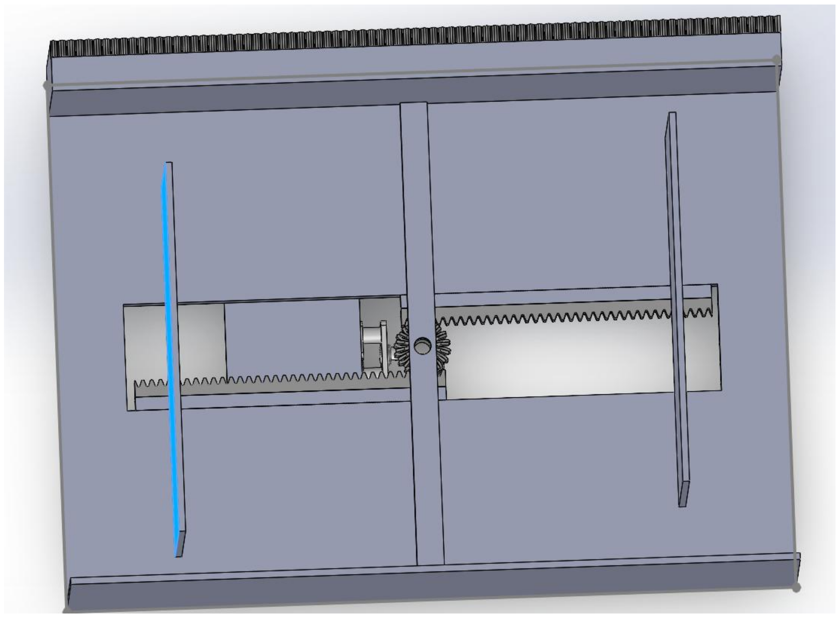
The system will move the tray to the other side, causing the rest of the cards to fall into the other side



After all of this, the lift will bring these two piles to the top where gear racks on the tray will split the cards apart.





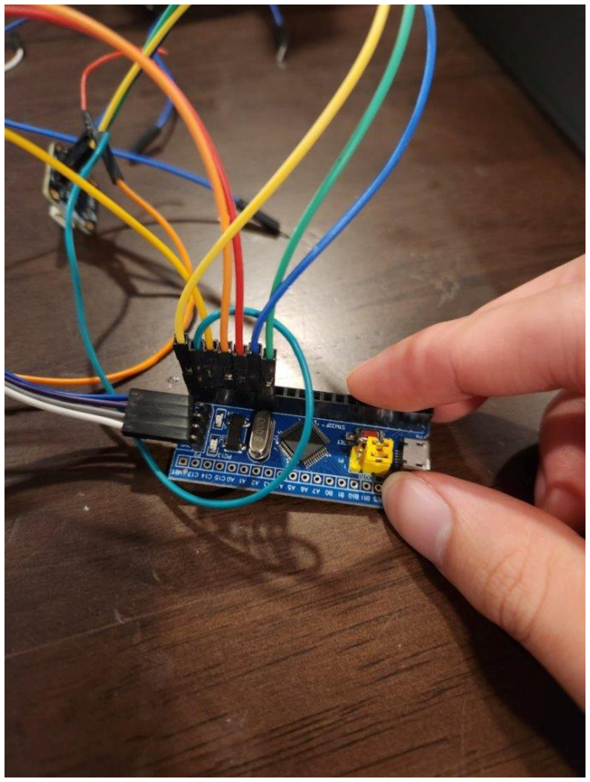


Thus completing the recirculation of the cards. This will occur 7 times or until the user is happy, after which the cards will be all shuffled properly in a pile in the middle.

After we selected this design, we 3d printed all the parts and I began working on the electrical side. This was also the point at which my capstone project came to an end and I was back to working on this by myself.

For the electrical side, I went with small n20 motors with low gear ratios for high torque. These motors were very small and quiet, perfect for the task and capable of moving the cards the distances required. Instead of using an ultrasonic sensor, this time I went for a time of flight sensor, although more expensive, it was much smaller and was far more accurate than the ultrasonic sensor which even had a minimum distance of 4cm. I decided to remove the color sensor, as after a lot of playtesting with the prototype, I noticed that no cards were ever coming out damaged, meaning that my product had no problem with shuffling unsleeved cards, therefore the sensor was unnecessary. The form factor of the smaller unsleeved card was addressed with the newly designed thickness adjusting system which could guarantee that no matter the thickness of the card, only one card would be dispensed at a time by the fly wheel.

To reduce the overall size of the product, I switched to using an STM32 Bluepill Microcontroller.



The pin headers were soldered on for easier testing and connections with the wires.

The final electrical system involved 6 motors, connected to DRV8833 motor drivers in pairs, in total taking up 6 GPIO pins on the Bluepill. Another 4 pins were connected to the time of flight sensor, 2 of which using i2c connection (SDA and SCL). I powered this electrical system with a powerbank I purchased on Amazon and the system worked to great success.

The next steps for this project, which I am currently working on is merging the 3d prints with the electrical system and running full tests with the completely assembled product. I plan to patent the product as well, especially the recirculation system, seeing as how there is nothing like this on the market currently. Once everything is complete, I plan to test it at local game stores and see how well it performs and also gain feedback.

Self-Stabilizing Airflow Tube (In Progress)

This is a project that I have been working on in a team that is in very early stages. Our goal has been to apply our collective knowledge in controls engineering to design a self-stabilizing system using controls principles. We circulated around many ideas before finally settling on one which would utilize fluid mechanics.

(sorry for no pictures just yet, please check my website regularly for potential updates on the project, https://www.joshuawuportfolio.info)

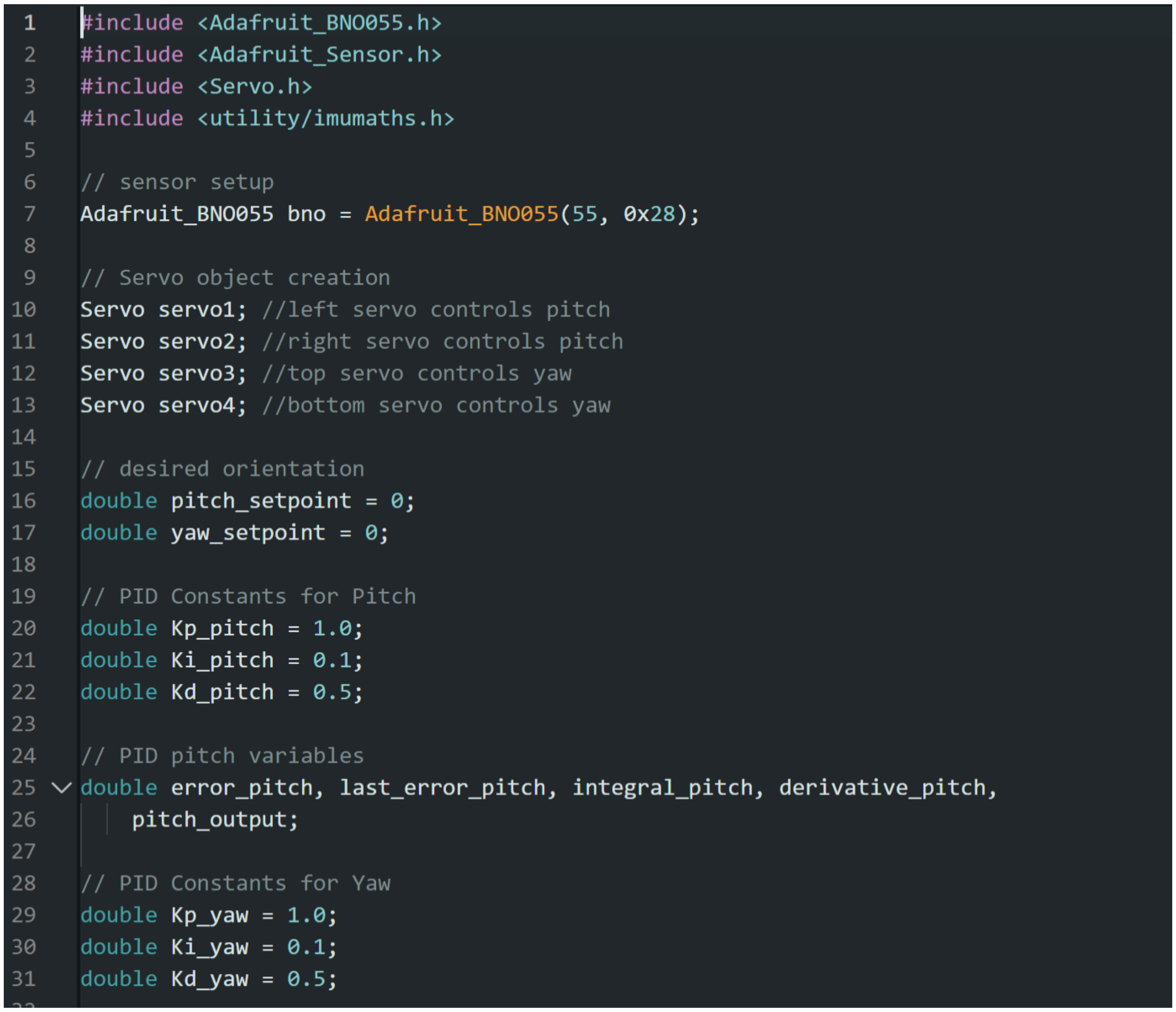
Our idea was to create a system that would be able to use lift force to adjust its position and stabilize itself when air is blown at it. To achieve this, we implemented 4 servo motors to control 2 airfoils that could control the airflow tube's pitch and yaw. Then, with an IMU, we would measure the angle of the tube which would in turn adjust the servos to level out the tube until it reached a stable equilibrium.

My role on the team has been working on the electrical and firmware side of the project.

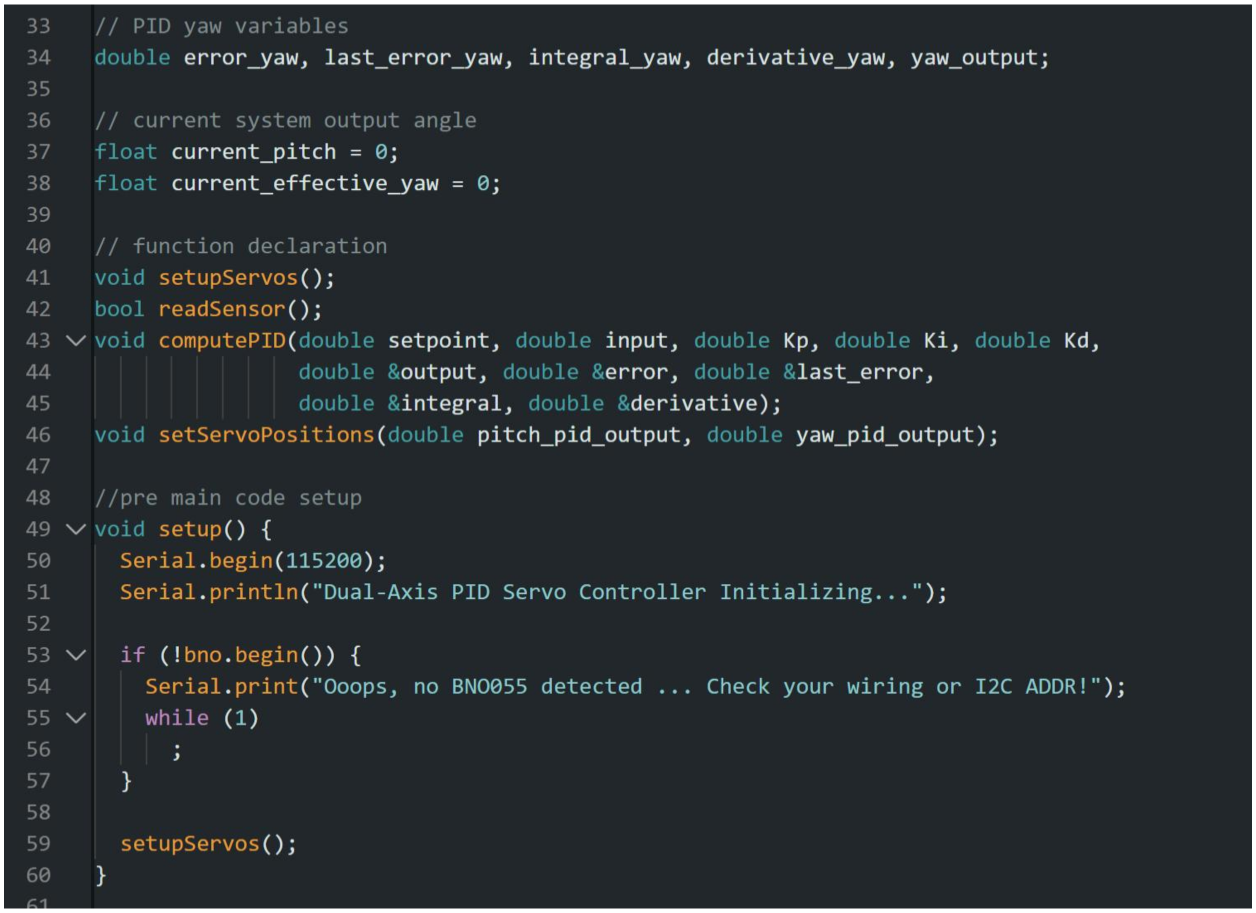
Personally I have designed the electrical circuit system which currently includes n IMU, 4 servo motors, and an Arduino Uno. The system receives power from an external battery.

(System uses 4 sg90 servos, Arduino Uno, BNO055 Absolute Orientation IMU)

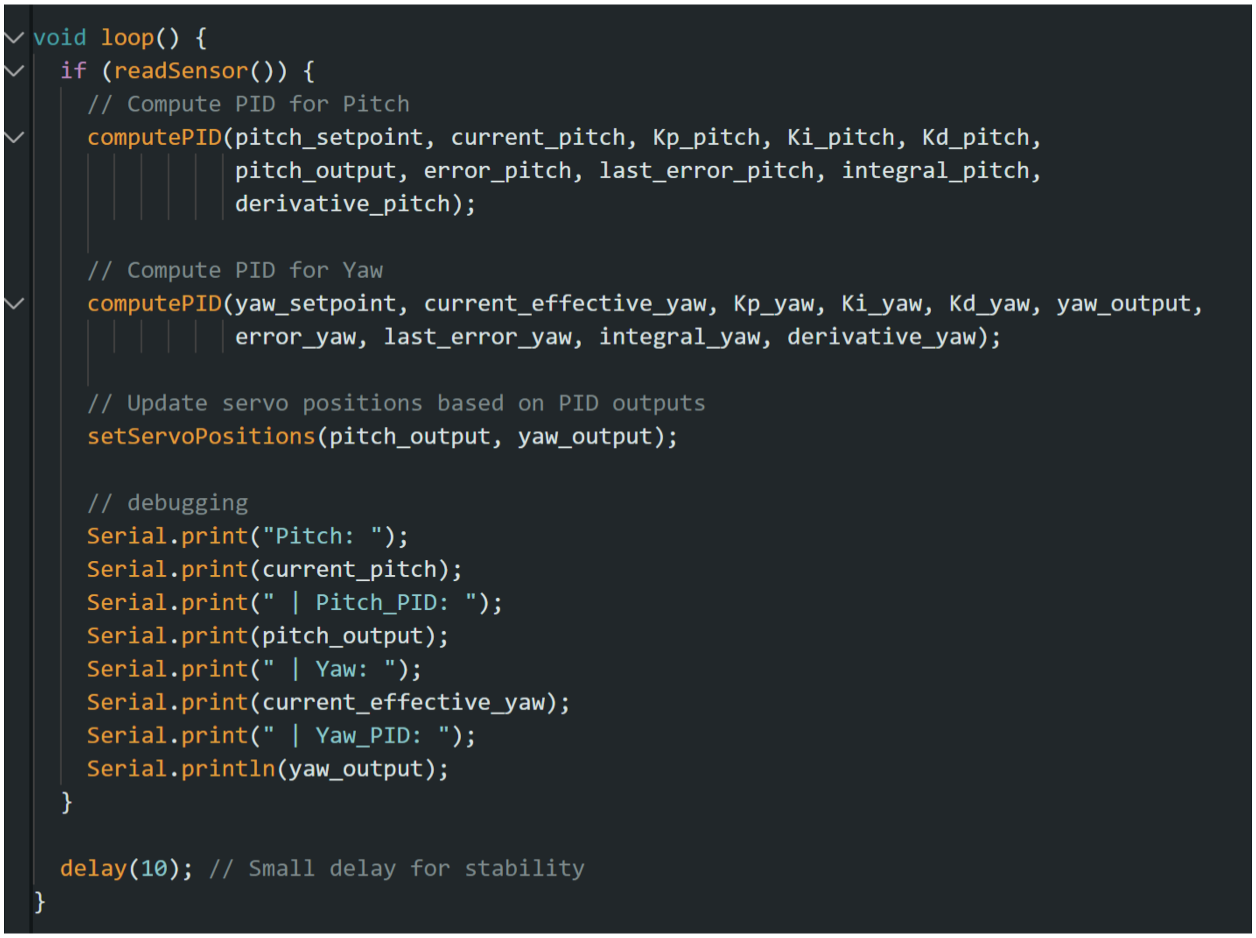
After setting this up, using the Arduino IDE, I have programmed firmware in C++ which successfully creates a PID controller for the system, altering servo angles depending on what angle the IMU measures. Below is an image of the code.



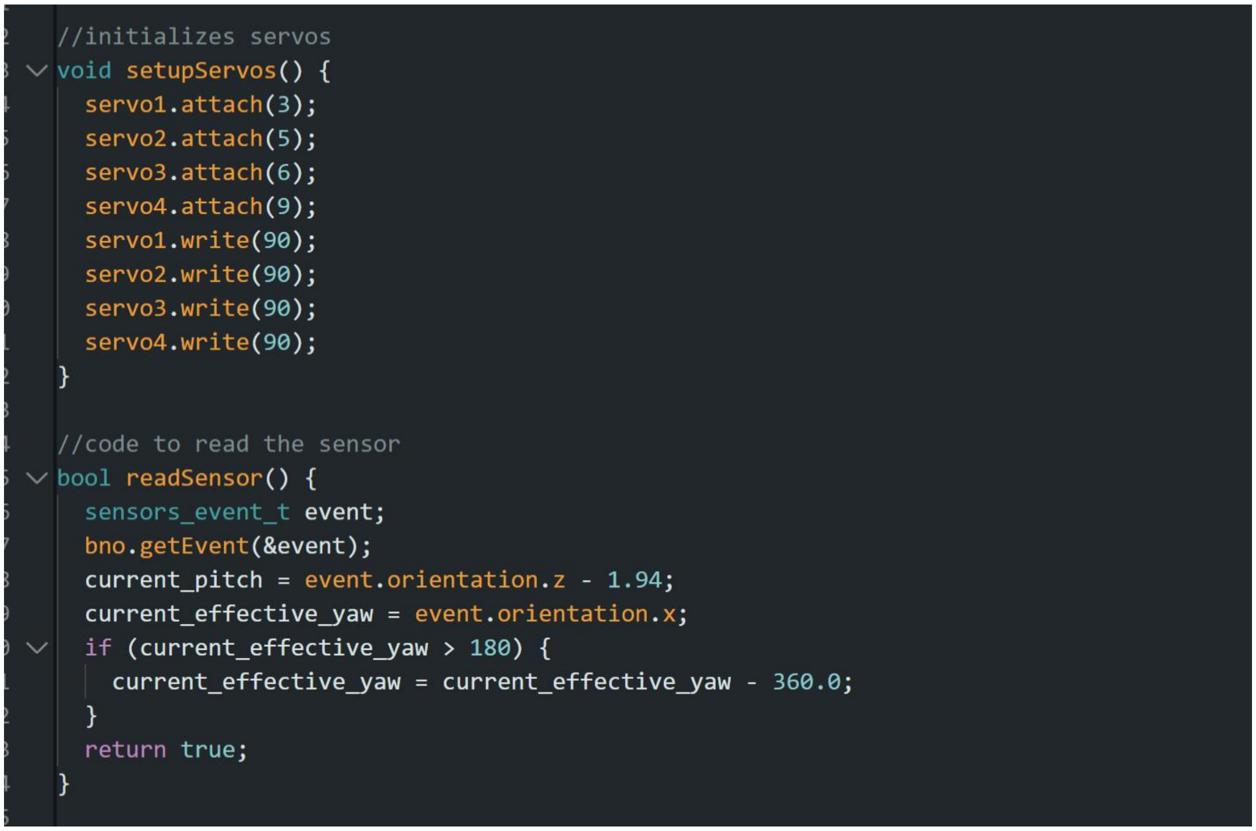
```
#include <Adafruit_BNO055.h>
#include <Adafruit_Sensor.h>
#include <Servo.h>
#include <utility/imumaths.h>

// sensor setup
Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28);

// Servo object creation
Servo servo1; //left servo controls pitch
Servo servo2; //right servo controls pitch
Servo servo3; //top servo controls yaw
Servo servo4; //bottom servo controls yaw

// desired orientation
double pitch_setpoint = 0;
double yaw_setpoint = 0;

// PID Constants for Pitch
double Kp_pitch = 1.0;
double Ki_pitch = 0.1;
double Kd_pitch = 0.5;

// PID pitch variables
double error_pitch, last_error_pitch, integral_pitch, derivative_pitch,
    pitch_output;

// PID Constants for Yaw
double Kp_yaw = 1.0;
double Ki_yaw = 0.1;
double Kd_yaw = 0.5;
```



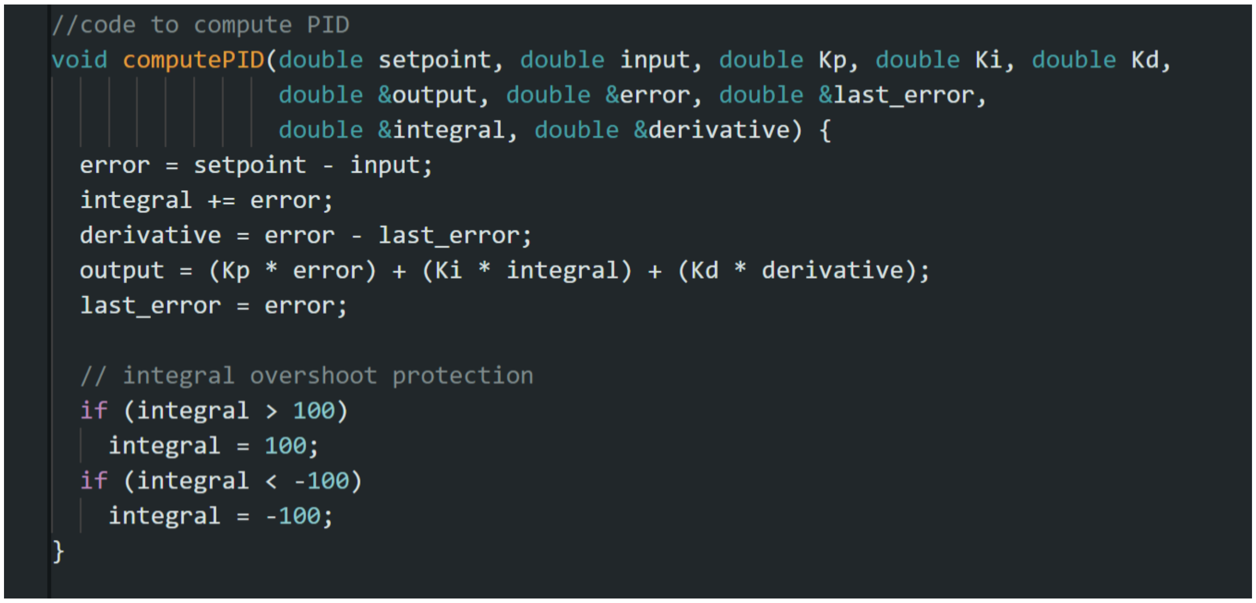
```cpp
// PID yaw variables
double error_yaw, last_error_yaw, integral_yaw, derivative_yaw, yaw_output;

// current system output angle
float current_pitch = 0;
float current_effective_yaw = 0;

// function declaration
void setupServos();
bool readSensor();
void computePID(double setpoint, double input, double Kp, double Ki, double Kd,
                double &output, double &error, double &last_error,
                double &integral, double &derivative);
void setServoPositions(double pitch_pid_output, double yaw_pid_output);

//pre main code setup
void setup() {
  Serial.begin(115200);
  Serial.println("Dual-Axis PID Servo Controller Initializing...");

  if (!bno.begin()) {
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while (1)
      ;
  }

  setupServos();
}
```



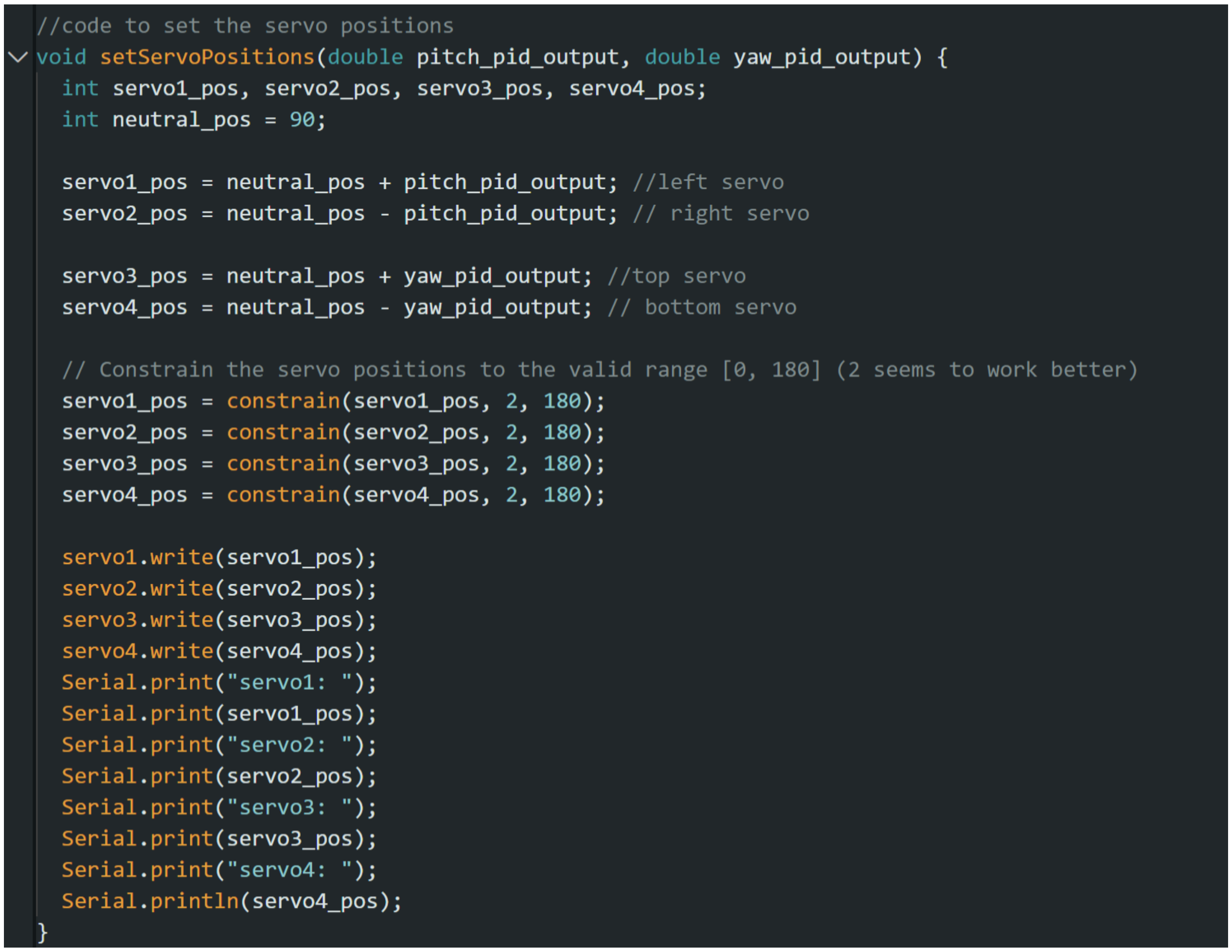
```
void loop() {
  if (readSensor()) {
    // Compute PID for Pitch
    computePID(pitch_setpoint, current_pitch, Kp_pitch, Ki_pitch, Kd_pitch,
               pitch_output, error_pitch, last_error_pitch, integral_pitch,
               derivative_pitch);

    // Compute PID for Yaw
    computePID(yaw_setpoint, current_effective_yaw, Kp_yaw, Ki_yaw, Kd_yaw, yaw_output,
               error_yaw, last_error_yaw, integral_yaw, derivative_yaw);

    // Update servo positions based on PID outputs
    setServoPositions(pitch_output, yaw_output);

    // debugging
    Serial.print("Pitch: ");
    Serial.print(current_pitch);
    Serial.print(" | Pitch_PID: ");
    Serial.print(pitch_output);
    Serial.print(" | Yaw: ");
    Serial.print(current_effective_yaw);
    Serial.print(" | Yaw_PID: ");
    Serial.println(yaw_output);
  }

  delay(10); // Small delay for stability
}
```



```
//initializes servos
void setupServos() {
  servo1.attach(3);
  servo2.attach(5);
  servo3.attach(6);
  servo4.attach(9);
  servo1.write(90);
  servo2.write(90);
  servo3.write(90);
  servo4.write(90);
}

//code to read the sensor
bool readSensor() {
  sensors_event_t event;
  bno.getEvent(&event);
  current_pitch = event.orientation.z - 1.94;
  current_effective_yaw = event.orientation.x;
  if (current_effective_yaw > 180) {
    current_effective_yaw = current_effective_yaw - 360.0;
  }
  return true;
}
```



```
//code to compute PID
void computePID(double setpoint, double input, double Kp, double Ki, double Kd,
                double &output, double &error, double &last_error,
                double &integral, double &derivative) {
  error = setpoint - input;
  integral += error;
  derivative = error - last_error;
  output = (Kp * error) + (Ki * integral) + (Kd * derivative);
  last_error = error;

  // integral overshoot protection
  if (integral > 100)
    integral = 100;
  if (integral < -100)
    integral = -100;
}
```



```
//code to set the servo positions
void setServoPositions(double pitch_pid_output, double yaw_pid_output) {
  int servo1_pos, servo2_pos, servo3_pos, servo4_pos;
  int neutral_pos = 90;

  servo1_pos = neutral_pos + pitch_pid_output; //left servo
  servo2_pos = neutral_pos - pitch_pid_output; // right servo

  servo3_pos = neutral_pos + yaw_pid_output; //top servo
  servo4_pos = neutral_pos - yaw_pid_output; // bottom servo

  // Constrain the servo positions to the valid range [0, 180] (2 seems to work better)
  servo1_pos = constrain(servo1_pos, 2, 180);
  servo2_pos = constrain(servo2_pos, 2, 180);
  servo3_pos = constrain(servo3_pos, 2, 180);
  servo4_pos = constrain(servo4_pos, 2, 180);

  servo1.write(servo1_pos);
  servo2.write(servo2_pos);
  servo3.write(servo3_pos);
  servo4.write(servo4_pos);
  Serial.print("servo1: ");
  Serial.print(servo1_pos);
  Serial.print("servo2: ");
  Serial.print(servo2_pos);
  Serial.print("servo3: ");
  Serial.print(servo3_pos);
  Serial.print("servo4: ");
  Serial.println(servo4_pos);
}
```

The current PID controller is likely to see significant change once we have derived the real mathematical representation for the system.

Current tasks within the project:

Applying Fluid Mechanics Principles to derive the mathematical model for the control system

Modeled and fabricated the test setup including a suspended tube with adjustable airfoil mounts